TAMPEREEN TEKNILLINEN YLIOPISTO

JUHA HIETANEN

IT-KALUSTON JA OHJELMISTOLISENSSIEN HALLINTAJÄRJESTELMÄ PK-YRITYKSESSÄ

Diplomityö

Tarkastaja: professori Tarja Systä
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekuntaneuvoston
kokouksessa 8. kesäkuuta 2011

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
HIETANEN, JUHA: IT-kaluston ja ohjelmistolisenssien hallintajärjestelmä PK-yrityksessä
Diplomityö, 44 sivua, 2 liitesivua
Tammikuu 2012
Pääaine: Ohjelmistotuotanto
Tarkastaja: professori Tarja Systä
Avainsanat: laitteet, lisenssit, tietojärjestelmä, tietokanta, laskutus

Tässä diplomityössä keskityttiin IT-laitteiden ja ohjelmistolisenssien hallintajärjestelmän suunnitteluun ja toteutukseen liittyviin yksityiskohtiin. Työn tavoitteena oli nykyaikaistaa Insinööritoimisto Comatec Oy:n vanhat kirjanpitojärjestelmät vastaamaan paremmin nykyajan vaatimuksia.

Yrityksen vanhat kirjanpitojärjestelmät perustuivat Excel-taulukkolaskentaohjelmalla tehtyihin listauksiin. Niiden ongelmina olivat ennen kaikkea laitteiden ja ohjelmistolisenssien tilatietojen historian puute sekä tiedostopohjaisuudesta johtuva yhden yhtäaikaisen käyttäjän rajoite. Tilatieto kertoo esimerkiksi laitteen tapauksessa sen, kenellä laite on käytössä, vai onko se mahdollisesti vapaana hyllyssä. Raportointi on nykyään yritysten toiminnassa tärkeässä osassa, ja Excel-pohjaisten vanhojen kirjanpitojärjestelmien heikkoutena onkin myös raportointimahdollisuuksien vähyys.

Toimeksiannon mukaan vanhat kirjanpitojärjestelmät oli uudistettava vastaamaan nykyajan tarpeita. Ensin kartoitettiin markkinoilla olevat valmiit järjestelmät ja tutkittiin niiden sopivuus tämän sovelluskohteen tarpeeseen. Mikään valmis järjestelmä ei tarjonnut sekä IT-laitteiden että ohjelmistolisenssien yhtäaikaista kirjanpitomahdollisuutta.

Seuraavaksi siirryttiin oman kirjanpitojärjestelmän suunnitteluun ja toteutukseen. Suunnittelutyön aluksi selvitettiin vielä tarkemmin uuden järjestelmän vaatimuksia. Selvitykset tehtiin tutkimalla vanhoja kirjanpitoja ja haastattelemalla IT-henkilöitä. Vaatimusten pohjalta voitiin lopputyö jakaa kolmeen osaan: uuden tietokannan suunnitteluun, vanhojen laitetietojen siirtoon uuteen tietokantaan sekä käyttöliittymän toteutukseen.

Uuden tietokannan suunnittelussa pyrittiin noudattamaan alalle vakiintuneita käytäntöjä, joista tehtiin aluksi kirjallisuusselvitystä. Tuloksena saatiin tietokantaskriptejä, joilla voitiin luoda tietokanta muutamassa sekunnissa. Vanhojen laitetietojen siirrossa uuteen tietokantaan käytettiin tietokantapalvelimen tarjoamia integrointityökalumahdollisuuksia. Käyttöliittymän toteutus vei työstä suurimman ajan.

Työn tuloksena luodusta IT-laitteiden ja ohjelmistolisenssien hallintajärjestelmästä tuli hyvin dokumentoitu kokonaisuus. Järjestelmä on nykyaikainen ja se korjaa aiempien kirjanpitojärjestelmien puutteet. Uusi järjestelmä täyttää sille asetetut vaatimukset ja luo hyvän pohjan raportoinnin kehittämiseen. Uusi IT-laitteiden ja ohjelmistolisenssien hallintajärjestelmä on selvästi parempi verrattuna vanhoihin kirjanpitojärjestelmiin, ja onkin suositeltavaa, että se otetaan käyttöön heti.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Signal Processing and Communications Engineering
HIETANEN, JUHA: Information System for Managing IT-devices and Software Licenses in Small and Medium Enterprises
Master of Science Thesis, 44 pages, 2 Appendix pages
January 2012
Major: Software Engineering
Examiner: Professor Tarja Systä
Keywords: Devices, Licenses, Information System, Database, Billing

This thesis focuses on design and implementation related details of a management system for IT devices and software licenses. The main objective was to modernize Engineering Office Comatec Oy's existing bookkeeping system to better meet today's standards.

The company's previous bookkeeping systems were based on Excel spreadsheet files. Their main shortcomings were the lack of status history of the information related to the devices and software license and the one simultaneous user limit that was due to the file-based systems that were used. The device status provides information about the device users or whether the device is potentially available. Reporting has a big role in companies' business today. The Excel-based bookkeeping systems have major weaknesses in reporting capabilities.

The modernization process was started by mapping the systems available on the market and their suitability to the needs of this specific application. It turned out that none of the market ready systems offer bookkeeping support for both IT devices and software licenses.

The next step was to design and implement our own bookkeeping system. The first part of the design was to find out the detailed requirements for the new system. This was done by studying old bookkeeping systems and interviewing IT people. Based on the requirements, the rest of the work was divided into the three parts: designing a new database, transferring the old device data to the new database, and user interface implementation.

The new database was designed relying on common practices of the industry. The result was a database script which could be used to create a database within seconds. The old device data was transferred to the new database using an integration tool offered by the database server. The implementation of the user interface was the most time consuming task.

The created management system for IT devices and software licenses was well documented. The system is modern and it addresses the shortcomings of the previous bookkeeping systems. The new system meets its requirements, and it creates a good foundation for the development of reporting. The new management system for IT devices and software licenses will bring a clear improvement compared to the old bookkeeping system and it is recommended to implement it as soon as possible.

ALKUSANAT

Tämä opinnäytetyö on tehty diplomi-insinööritutkinnon lopputyönä Ohjelmistotekniikan laitokselle Tampereen Teknillisellä yliopistolla. Työn aihe on peräisin Insinööritoimisto Comatec Oy:n käytännön projektista, jossa kehitettiin yrityksen IT-kaluston ja ohjelmistolisenssien hallintajärjestelmä vastaamaan paremmin nykyajan tarpeita. Kiitos Comatecille ja toimitusjohtaja Aulis Asikaiselle, että sain tehdä heille diplomityön mielekkäästä ja kiinnostavasta aiheesta.

Comatecin puolesta työn tekemistä ovat ohjanneet esimieheni IT-päällikkö Mauno Jokinen sekä työtoverini IT-päällikkö Juha Marjanen, joiden kanssa olemme keskittyneet yhteisissä istunnoissamme lähinnä asiasisällön paikkansapitävyyteen ja tietokantarakenteen riittävyyteen. Kiitos heille molemmille uhraamastaan ajasta ja kärsivällisyydestä, jota ovat yhteisissä palavereissamme osoittaneet. Maunolle kuuluu lisäksi suuri kiitos työn jouhevasta yhdistämisestä päivätyöni kanssa ja ymmärtävästä kannustamisesta työn pitkittyessä.

Alun perin työni ohjaajana piti Ohjelmistotuotannon laitokselta olla suuresti arvostamani professori Ilkka Haikala, jonka kanssa pidimme jo yhteisen aloitusistunnon työn tekemisestä. Työn käytännön osuuden suorittamisen myötä yhteydenpitoni laitokselle jäi alkutaipaleella vähemmälle. Myöhemmin käydessäni laitoksella huomasin Ilkan huoneen olevan tyhjillään, ja ihmettelin, ettei allekirjoittaneelle ollut ilmoitettu asiasta mitenkään. Kysyin toiselta professorilta ja sain tietää Ilkan kuolleen muutama kuukausi aloituspalaverin jälkeen. Olin surullinen ja järkyttynyt. Ilkan myötä menetimme rautaisen asiantuntijan, suuren luennoitsijan ja todella mukavan ihmisen. Olen todella kiitollinen niistä ohjelmistotuotannon kursseista, joille sain osallistua vielä Ilkan aikana. Aina oli mukava mennä Ilkan luennoille ja ne olivat ehdottomasti opintoviikkojen kohokohtia. Ilkan ohjaamana valmistui yli 900 diplomityötä, joiden joukkoon tämä työ ei enää valitettavasti ehtinyt.

Minulle osoitettiin hyvin pian uusi ohjaaja, professori Tarja Systä. Olen erittäin kiitollinen Tarjalle hänen joustavasta ja ammattitaitoisesta ohjauksestaan. Kommentteja on kolahtanut sähköpostilaatikkoon esimerkiksi lauantain ja sunnuntain välisenä yönä. Olin erittäin etuoikeutetussa tilanteessa saadessani työskennellä hänen ohjauksessaan. Kiitos Tarja!

Työn kuluessa allekirjoittaneen korkeaa työmotivaatiota ovat ylläpitäneet rakkaimmat läheiseni: vaimoni Elina sekä 27.11.2011 syntynyt esikoispoikamme Mico Kalevi. Olen suuren kiitoksen velkaa Elinalle hänen kärsivällisyydestään ja kannustuksestaan diplomityön valmistumisen kestäessä.

Lopuksi haluan vielä kiittää Jumalaa, jonka avulla mahdoton muuttuu mahdolliseksi.

Ulvilassa 18.12.2011,
Juha Hietanen

SISÄLLYS

LYHENTEET JA MERKINNÄT

Termi	Selite
.NET Framework	.NET Framework on Microsoftin kehittämä ohjelmistokomponenttikirjasto, jota Microsoftin VisualStudio.NET -ympäristössä kehitetyt ohjelmistot käyttävät.
Access	Access on Microsoftin kehittämä tietokantojen hallintaohjelma, joka kuuluu Microsft Office -ohjelmistotuoteperheeseen.
ADO.NET	ADO.NET on Microsoftin kehittämä tietokantojen ohjelmointirajapinta, joka tukee sekä relaatiomallia että XML-kieltä. Se on osa .NET -ohjelmistokomponenttikirjastoa.
Business Intelligence Development Studio, BIDS	Microsoftin kehittämä työkalu muun muassa raporttiprojektien (SRSS) ja integrointiprojektien (SSIS) luomiseksi.
C, C++	C ja C++ ovat ohjelmointikieliä, joihin tässä työssä käytetty C# osaksi perustuu.
C#	C# on ohjelmointikieli, joka on osa .NET -konseptia. Se kehitettiin Microsoftin toimesta muun muassa yhdistämään C++:n tehokkuus ja Java-kielen helppokäyttöisyys.
Common Language Runtime, CLR	Common Language Runtime on ajonaikainen käyttöympäristö, jossa .NET -ohjelmia suoritetaan.
Data	Data on merkkejä tai symboleja, jotka muodostavat informaatiota, jos niille annetaan merkitys.
Debuggaus	Debuggaus on ohjelmistotuotannon osa, jossa testauksessa löytyneen virheellisen toiminnan aiheuttanut virhe paikallistetaan ja korjataan.
Delphi	Toinen nimitys Object Pascal -ohjelmointikielelle. Tunnetaan parhaiten Borland Delphi -kehitysympäristön alkuperäisenä ohjelmointikielenä.

Termi	Selite
Excel	Taulukkolaskentaohjelma, joka on tarkoitettu numeerisen datan käsittelyyn. Data on tallennettu kaksiulotteisen taulukon soluihin. Excel on kehitetty Microsoftin toimesta ja se on osa Microsoft Office -ohjelmistotuoteperhettä.
IBM	IBM on iso yritys, joka on parhaiten tunnettu suurtietokoneiden ja palvelinten valmistamisesta. Yritys on perustettu jo vuonna 1988.
IT-laitteet, ICT-laitteet	Käsittää kaikki informaatioteknologiaan liittyvät laitteet, muun muassa keskusyksiköt, kannettavat tietokoneet ja hiiret. ICT-käsite pitää sisällään myös kommunikaatiolaitteet, kuten puhelimet.
Informaatio	Informaatio on sellaista dataa, johon on liitetty tai johon on liitettävissä jokin merkitys tai tulkinta, merkkijonon ilmaisema viesti. Kun sille on annettu merkitys ja se sisäistetään, siitä tulee tietoa.
Integrated Development Environment, IDE	Ohjelmointiympäristö, jolla ohjelmoija toteuttaa ohjelmiansa. Yksinkertaisimmillaan ohjelmointiympäristö on tekstieditori ja ohjelmointikielen kääntäjä.
Internet	Internet on maailmanlaajuinen tietoverkko, joka yhdistää paikallisia tietoverkkoja toisiinsa. Internet on avoinna kaikille, jotka haluavat liittyä siihen noudattamalla sen teknisiä viestintäsääntöjä (protokollia).
IT Inventory Manager, ITIM	Tässä työssä luotavan IT-kaluston ja lisenssien hallintajärjestelmän nimi. Samalla nimellä kutsutaan niin uutta tietokantaa kuin myös luotavaa käyttöliittymää.
ISO	ISO (International Organization for Standardization) on kansainvälinen standardoimisorganisaatio, joka on perustettu vuonna 1947. Se tuottaa kansainvälisiä standardeja.

Termi	Selite
Kustannuspaikka, tulospaikka	Yrityksen sisäinen yksikkö, joka koostuu yrityksen työntekijöistä; kustannuspaikan esimiehestä ja hänen alaisistaan.
Leasing, liisaus	Tarkoittaa käyttöomaisuuden pitkäaikaista vuokrausta. Usein liisataan muun muassa autoja ja koneita esimerkiksi 3-5 vuoden sopimuksella.
Lisenssi	Tietokoneohjelmiston käyttöoikeutta sanotaan lisenssiksi. Se voi olla numerosarja, tietokonetiedosto, konkreettinen laite tai jokin edellisten yhdistelmä. Konkreettinen laite voi olla esimerkiksi muistitikun tapainen USB-porttiin kytkettävä laite. Myös verkkolisenssi on yksi lisenssimuoto, tällöin lisenssi on lisenssipalvelimelta varattava lupa ohjelman käyttöön.
Lisenssipalvelin	Palvelintietokone, joka ylläpitää yrityksen käytössä olevien verkkolisenssien kirjanpitoa. Kukin lisenssi oikeuttaa yhden ohjelman käyttämiseen. Kaikkien lisenssien ollessa käytössä on odotettava, kunnes joku ohjelman avannut sulkee ohjelman ja tällä tavoin vapauttaa lisenssin jonkun toisen käyttöön.
Lookup	Lookup-termillä tarkoitetaan tietokantojen tapauksessa ennalta määriteltyä listaa joistakin objekteista. Esimerkiksi voidaan määritellä, että lisenssi on tyypiltään joko verkko tai kiinteä.
Notepad++	Notepad++ on avoimeen lähdekoodiin pohjautuva tekstieditori Windows-käyttöjärjestelmille. Perinteisestä tekstieditorista sen erottaa monipuolisuus.
Oliotietokanta	Olio-ohjelmoinnin yleistyessä myös olioita tallentavia tietokantoja on kehitetty. Oliotietokannat ovat relaatiokantoja helpompikäyttöisiä olio-ohjelmointia käytettäessä, mutta niiden haittapuolena on raakadatan tarkastelun haastavuus.

Termi	Selite
Palvelin, server	Palvelimella tarkoitetaan tietokonetta, joka hoitaa vain ennalta määritettyä tehtävää. Tehtävä voi olla esimerkiksi jonkin sovelluksen suorittaminen, sähköpostin välitys tai ohjelmavalmistajan lisenssikirjanpidon ylläpito (lisenssipalvelinohjelmiston suoritus). Nimitys tulee siitä, että kyseinen palvelintietokone palvelee muita tietokoneita.
Relaatiotietokanta	Edelleen hyvin suosittu tietokantamalli, jossa tieto tallennetaan pääosin selkokielisenä tekstinä tauluihin. Tauluilla on keskenään suhteita, siis relaatioita.
Scrum	Scrum on ketterä ohjelmistokehitysmalli.
Skripti	Skriptillä tarkoitetaan komentosarjaa, joka suorittaa useita komentoja peräkkäin. Komennot voivat olla esimerkiksi tietokantakomentoja tai käyttöjärjestelmän komentoja.
Skriptitiedosto	Skriptitiedostolla tarkoitetaan tekstimuodossa tallennettua tiedostoa, joka sisältää skriptin. Skripti suoritetaan kerralla kokonaan ja se voidaan helposti toistaa kerta toisensa jälkeen.
SQL	SQL (Structured Query Language) on IBM:n kehittämä standardoitu kyselykieli, jolla relaatiotietokantoihin voidaan tehdä hakuja, muutoksia ja lisäyksiä. Lähes kaikki relaatiotietokannat ymmärtävät SQL-kieltä.
SQL Server, MSSQL	SQL Server on Microsoftin kehittämä relaatiotietokantajärjestelmä, jota hyödynnetään tämän projektin toteutuksessa.
SQL Server Integration Services, SSIS	SQL Server -tietokannan lisäosa, jolla pystytään helposti toteuttamaan tietomassojen yhdistämiseen liittyviä rutiineja ja jopa automatisoimaan niitä.
SQL Server Management Studio	Microsoftin SQL Server -tietokannanhallintaohjelma Windowsille.
SQL Server Reporting Services, SRSS	Microsoftin luoma raportointialusta. Raportit voidaan muodostaa SQL-palvelimelta haetusta informaatiosta.

Termi	Selite
Surrogaatti	Tietokannan taululle keinotekoisesti keksitty pääavain tai tunniste (ID). Usein juokseva kokonaisluku nollasta eteenpäin.
Tieto	Tieto on tulkittua ja sisäistettyä informaatiota. Filosofien perinteisen määritelmän mukaan tieto on hyvin perusteltu tosi käsitys.
Tietokanta	Tietokanta on tietokannanhallintajärjestelmällä hallittava joukko loogisesti toisiinsa liittyvää tietoa.
Tietokannanhallintajärjestelmä	Tietokannanhallintajärjestelmä edustaa tietokantaa ja tarjoaa ainoan sallitun mahdollisuuden käsitellä tietokannan tietoja.
Tietokantaskripti, tietokantaskriptitiedosto	Tietokantaskriptillä tarkoitetaan skriptitiedostoa, joka suoritetaan tietokannanhallintaohjelmassa. Tietokantaskriptillä voidaan muun muassa luoda tietokanta kaikkine tauluineen ja relaatioineen sekä asettaa oikeudet ja lisätä mahdollinen kiinteä informaatio tauluihin.
Tietokoneohjelmisto	Koostuu yhdestä tai useasta tietokoneohjelmasta, kaikkien ohjelmien käyttämistä tiedostoista ja koko ohjelmistotuotteeseen liittyvästä dokumentaatiosta.
Tietokonepaketti, konepaketti	Useasta laitteesta koostuva kokonaisuus, jossa on esimerkiksi keskusyksikön lisäksi näppäimistö, hiiri ja kaksi näyttöä. Myös kannettavien tietokoneiden tapauksessa voidaan puhua konepaketeista; niihin kuuluu itse kannettavan tietokoneen lisäksi esimerkiksi telakka, näppäimistö, kaksi hiirtä ja ulkoinen näyttö.
Unified Modeling Language, UML	UML on vuonna 1997 standardoitu graafinen mallinnuskieli, joka sisältää 13 eri kaaviotyyppiä. Tässä työssä käytetään rakennekuvaukseen tarkoitettua luokkakaaviota ja käyttäytymiskuvaukseen tarkoitettua käyttötapauskaaviota. Lisäksi tapahtumasekvenssikaavio kuvaa eri tapahtumia aikajanalla.

Termi	Selite
VB.NET	VB.NET on Microsoftin kehittämä ohjelmointikieli, joka perustuu edeltäjäänsä Visual Basic -ohjelmointikieleen. VB.NET on luotu .NET -komponentin myötä.
Verkkolisenssi	Lähiverkon kautta palvelimelta haettava lisenssi, jota voidaan helposti ja joustavasti jakaa halutulle käyttäjäryhmälle.

1. JOHDANTO

1.1. Diplomityön tausta

Tiedon määrä maailmassa lisääntyy nopeasti, ja tiedon hallinnan merkitys korostuu päivä päivältä. Pelkkä suuri datamäärä ei ole hyödyllistä juuri muille kuin datan säilömiseen erikoistuneille yrityksille.

Datalle on annettava merkitys, jotta siitä saadaan informaatiota. Tietoa informaatiosta tulee kun on varmistuttu, että se on perustellusti totta. Datan määrä tulisi pitää mahdollisimman pienenä, mikä parhaiten onnistuu jättämällä turha data huomiotta.

Isojen tietomäärien tallennuksessa kysymykseen tulee tietokanta, joka on tietokannanhallintajärjestelmällä hallittava joukko loogisesti toisiinsa liittyvää tietoa. Juuri tietojen loogiseen jäsentelyyn tulee kiinnittää erityistä huomiota, jotta vältyttäisiin toisteisuudelta, tieto olisi helposti käytettävissä ja mahdollinen jatkojalostus olisi helppoa.

Yrityksissä on monia erilaisia kirjanpitoja, jotka tarkoittavat loogisesti toisiinsa liittyviä joukkoja tietoa. Eräs merkittävä kirjanpito on yrityksen tieto- ja viestintätekniikkalaitteiden (ICT-laitteiden) kirjanpito, jolla pidetään yllä yrityksen omistuksessa olevia laitteita ja niiden käyttäjiä – vaikkakin yhä useammin laitteet omistaa rahoitusyhtiö ja laitteet ovat niin sanotusti leasing-laitteita. Nykyään on jopa sellaisia työntekijöitä, joilla on useampia tietokonepaketteja käytössään.

1.2. Diplomityön määrittely, tavoitteet ja rajaus

Tämän diplomityön aihe on peräisin Insinööritoimisto Comatec Oy:ltä. Toimeksiannon mukaan Comatecin IT-kaluston ja lisenssien hallintajärjestelmä oli modernisoitava. Toimeksiantoon kuului vastaavien markkinoilla olevien järjestelmien kartoitus ja niiden sopivuuden tutkiminen. Ellei markkinoilta löytyisi tehtävään soveltuvaa hallintajärjestelmää, olisi sellainen suunniteltava ja toteutettava itse.

Suunnittelutyössä tulisi kiinnittää erityistä huomiota selkeään ja yksinkertaiseen tietokantaratkaisuun. Samalla kun uusi tietokanta luotaisiin, olisi sinne siirrettävä vanhan laitekirjanpidon tiedot. Lisäksi vaatimuksena oli käyttöliittymän toteutus uuteen tietokantaan. Käyttöliittymän kautta pitäisi onnistua tietojen selailu, lisäys ja päivitys, muut toiminnot onnistunevat ainakin aluksi esimerkiksi suoraan tietokannanhallintajärjestelmän kautta käsityönä.

Alkuperäisen toimeksiannon mukaan ulkopuolelle oli rajattu lisenssit ja niihin liittyvät tiedot sekä toiminnallisuus. Hyvin pian suunnittelutyön alettua oli kuitenkin selvää, että myös lisenssit tuli ottaa mukaan, sillä ne linkittyvät erittäin vahvasti

laitteistoihin sekä myös IT-osaston laskuihin. Laitteet, lisenssit ja laskut muodostavat tavoitteeksi asetetun hallintajärjestelmän kolme pääosa-aluetta.

Myös käyttöliittymän käyttäjäryhmien osalta piti harkita rajausta hyvin pian työn alettua. Käyttäjäryhmillä oli tarkoitus jakaa käyttöliittymän toiminnallisuuksia eri henkilöiden tarpeisiin sen mukaan, mihin työntekijäryhmään he kuuluvat. Esimerkiksi IT-henkilöt pystyisivät tekemään enemmän asioita käyttöliittymän kautta kuin osastojen esimiehet, joiden oli myös alkuvaiheessa tarkoitus käyttää käyttöliittymää. Käyttäjäryhmien toteutus päätettiin kuitenkin siirtää myöhempään ajankohtaan ja pois tämän työn piiristä. Käyttäjäryhmätoiminnot tultaisiin suorittamaan osin jollakin toisella tekniikalla, kuin alun perin oli aiottu.

Mikäli päädyttäisiin oman järjestelmän suunnitteluun, tulisi se toimeksiannon rajausten mukaan suunnitella tallentamaan tietonsa Microsoft SQL Server -tietokantaan. Yrityksen kaikki merkittävät tietokannat on toteutettu juuri edellä mainitulla tietokantamoottorilla. Lisäksi toimeksiantaja halusi juuri relaatiotietokantajärjestelmän IT-hallintajärjestelmän alustaksi, jotta tallennetut tiedot olisivat selkokielisinä tauluissa ja niitä voitaisiin tarvittaessa tutkia ja jopa muokata käsin. Oliotietokannoissa vastaava käsin tutkiskelu ei olisi ollut niin yksinkertaista.

Uuden hallintajärjestelmän tietokannasta tuotetaan yrityksen sisäiseen käyttöön useita erilaisia raportteja, muun muassa osastojen esimiehille, yrityksen johdolle sekä etenkin talousosastolle. Raportit on tarkoitus tuottaa käyttäen SQL Server Reporting Services -palvelua (SRSS). SRSS:llä tuotettavat raportit eivät suunnittelun osalta kuuluneet työn piiriin laajuutensa ja suuren lukumääränsä vuoksi. Liitteessä 2 on kuitenkin esimerkkiraportti, jossa kuvataan osastokohtaista laiteraporttia. Raportti toimii Internet-selaimella ja siinä olevia osastoja voidaan selaimella aukaista ja sulkea. Esimerkin vuoksi ensimmäinen osasto on mallitulosteessa aukaistu.

1.3. Työn rakenne

Tämä diplomityö jakaantuu kahteen isompaan pääkokonaisuuteen. Ensimmäinen puolisko käsittää teoriaosan, joka keskittyy taustalla olevien teorioiden esittelyyn. Välissä on suunniteltavan ja toteutettavan järjestelmän lähtötiedot ja vaatimukset, joiden kautta siirrytään toiseen työn pääpuoliskoista. Toinen puolisko on soveltava osa, jossa käydään läpi työn käytännön työstämiseen liittyviä asioita. Lopuksi vedetään yhteen työn tulokset.

Luku 2 sisältää esittelyn työssä käytettävistä työkaluista, jotka on jaoteltu alilukuihin niiden käyttötarkoituksen perusteella.

Luvussa 3 esitellään työn suorittamiseen olennaisesti vaikuttaneet teoriat. Teorialuku jakaantuu kolmeen pääosaan: tietokantojen historian kuvaamiseen, relaatiotietokantojen suunnitteluprosessin läpikäyntiin sekä ohjelmistojen elinkaaren vaiheiden erittelyyn.

Luvussa 4 käydään läpi lähtökohtia tälle työlle. Luvussa kuvaillaan vanhaa järjestelmää sekä esitellään uuden hallintajärjestelmän pääosat.

Luvussa 5 esitellään työn kohteena olevien osa-alueiden vaatimuksia. Kukin osa-alue käydään erikseen läpi: tietokanta, integraatioprojekti sekä käyttöliittymä.

Luvussa 6 , kuinka työ konkreettisesti suoritettiin. Luvun ensimmäisessä aliluvussa kerrotaan muiden IT-hallintajärjestelmien vertailututkimuksesta, mutta muuten luku jakaantuu kolmeen alilukuun kunkin edellä mainitun osa-alueen mukaan.

Luku 7 keskittyy tulosten esittelyyn ja niiden analysointiin osa-alueittain. Viimeisessä aliluvussa arvioidaan työn onnistumista.

Luvussa 8 kuvataan työn aikana mieleen nousseita jatkokehitysajatuksia tulevaisuuden varalle. Jatkokehitysideat ovat isompia kokonaisuuksia ja ideoita, joita ei ollut mahdollista toteuttaa tämän työn puitteissa.

Luku 9 esittelee koko työn loppupäätelmät. Luvussa annetaan toimeksiantajalle suositukset, joilla saadaan työstä koko hyöty irti.

1.4. Yritysesittely

Insinööritoimisto Comatec Oy on perustettu vuonna 1986 ja se on viime vuosina kasvanut merkittävästi luonnollisen kasvun lisäksi yritysostojen kautta. Tällä hetkellä yrityksen palveluksessa on yli 320 henkilöä. Comatec on kone- ja laitesuunnitteluun erikoistunut asiantuntijaorganisaatio, jonka toimialoja ovat liikkuvat työkoneet, tuotantolaitteet, liikennevälineet ja teollisuusautomaatio.

Tampereella perustettu Comatec on laajentunut vuosien kuluessa useille paikkakunnille. Nykyään konsernin toimipisteitä on Tampereella, Vantaalla, Lahdessa, Turussa, Hämeenlinnassa, Joensuussa, Imatralla ja Varkaudessa. Uusi aluevaltaus tapahtui tänä vuonna, kun Comatec perusti toimiston Viroon.

Comatecin palvelut koostuvat ideointi- ja konseptipalveluista, projektin hoito- ja johtopalveluista sekä suunnittelu- ja asiantuntijapalveluista. Suunnittelupalvelut kattavat mekaniikka-, sähkö-, ja automaatiosuunnittelun. Asiantuntijapalveluita ovat muun muassa testaus, tekninen laskenta, tuoteturvallisuus sekä elinkaaripalvelut. Konserniin kuuluvat lisäksi painelaite- ja kattilasuunnitteluun erikoistunut Rantotek Oy sekä Insinööritoimisto Metso Oy.

Comatecin asiakkaita ovat teknologiateollisuusyritykset, jotka tuottavat koneita ja laitteita business to business -toimintaympäristössä. Yrityksen tuotteita ovat suunnittelupalvelut sekä asiantuntija- ja projektinhallintapalvelut. Comatecin asiakkaat hyötyvät yhtiön palveluista siten, että he saavat sovitun aikataulun mukaisesti käyttöönsä kustannustehokkaasti toteutettuja, viimeisteltyjä ja kestäviä suunnitteluratkaisuja, jotka palvelevat pitkäjänteisesti asiakkaan omia tuote- ja valmistusprosesseja. Comatecin oma tulos perustuu kattavaan markkinointiin, erinomaiseen tekniseen suunnittelutaitoon ja -kokemukseen, laadun käsitteen syvälliseen ymmärtämiseen sekä osuvaan, asiakas- ja toimialakohtaiseen hinnoitteluun ja pitkäjänteisiin asiakassuhteisiin.

Comatecin liikkuvien työkoneiden osaston asiakkaita ovat muun muassa Kalmar, Metso Minerals, John Deere, Patria, Sandvik Mining and Construction, Mantsinen,

KOME ja RCM Harvester. Tuotantolaitteiden osasto muodostaa toisen merkittävän kokonaisuuden, ja sen asiakkaita ovat muun muassa Corenso, KONE, Outokummun Metalli, Oilon, Outotec ja Areva. Liikennevälineiden osasto jakaantuu kiskokalustoon ja laivanrakennukseen, vastaavasti suurimmat asiakkaat ovat VR ja ABB Marine. Kattilalaitokset ovat tytäryhtiö Rantotekin erikoisosaamista ja heidän merkittävimpiä asiakkaitaan ovat Andritz, Foster Wheeler, Fortum, Metso Power, Rautaruukki, Sweco Industry, ÅF Process, Vapor Finland ja YIT Finland. [1]

Comatecin asiakkaiden, eli toimeksiantajien, tarpeet määrittävät käytettävät ohjelmistot. Koska asiakkaita on paljon, käytössä on useita kymmeniä eri ohjelmistoja monilta eri ohjelmistovalmistajilta ja ohjelmistoaloilta. Siitä syystä mukana on myös joitakin hyvinkin eksoottisia ohjelmia. Yritys toimii muunkin IT-infrastruktuurin osalta erittäin asiakaslähtöisesti, mistä seurauksena on sekalainen valikoima erilaisia IT-laitteita.

1.5. Tutkimusmenetelmät ja työn kieli

Tässä työssä käytettiin pääosin empiirisiä ja konstruktiivisia tutkimusmenetelmiä. Työn tavoitteeksi asetetun IT-laitteiden ja ohjelmistolisenssien hallintajärjestelmän vaatimuksia selvitettäessä käytettiin empiirisiä tutkimusmenetelmiä, kuten haastatteluja. Haastatteluilla kerättiin uuden hallintajärjestelmän vaatimuksia. Haastatteluista saatuja tietoja voidaan pitää laadullisena aineistona. Työssä ratkaistiin käytännön ongelma luomalla uusi konstruktio, joka tässä tapauksessa oli uusi IT-laitteiden ja ohjelmistolisenssien hallintajärjestelmä.

Diplomityön dokumentaatiokielenä käytettiin suomea, mutta käytännön toteutuksessa on pyritty käyttämään mahdollisuuksien mukaan englantia. Tämän vuoksi osissa kuvia saattaa esiintyä englanninkielisistä termistöä. Käytetyt työkalut ja työn sidosjärjestelmät ovat englanninkielistä, minkä vuoksi on luonnollista tuottaa ohjelmakoodi englanniksi. Englanninkielisyyttä pidetään standardikielenä ohjelmistoalalla.

2. TYÖKALUT

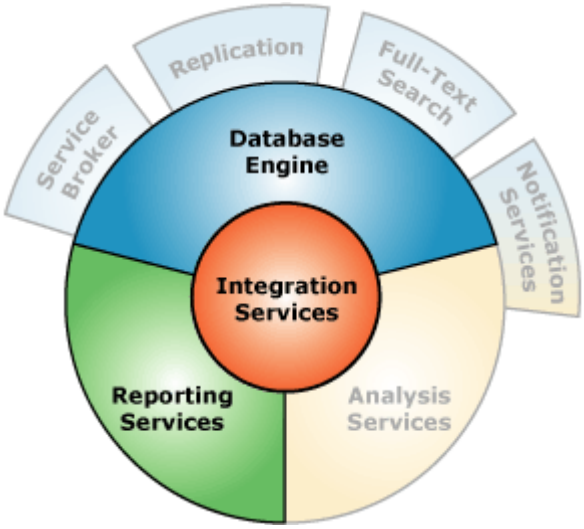
2.1. Yleistä työkaluista

Lähes kaikki työssä käytettävät työkalut ovat Microsoftin valmistamia, kuten koko se IT-infrastruktuuri, jossa työn tuloksena syntyvät järjestelmät tulevat toimimaan. Yhden valmistajan työkalujen käyttäminen ei ole välttämätöntä, mutta tässä tapauksessa lähtökohdat tukevat sitä, ja etuna on myös se, että todennäköisesti yhteensopivuusongelmia ei tule esiintymään.

2.2. Microsoft SQL Server 2005

Toimeksiantajayrityksessä on tietokantajärjestelmänä käytössä Microsoftin MS SQL Server 2005, johon pääosa yrityksen tietokantadatasta on tallennettu. Toimeksiannon mukaan oli itse suunniteltu tietokantajärjestelmä suunniteltava tallentamaan tiedot juuri MS SQL Server 2005 -tietokantajärjestelmään.

MS SQL Server 2005 -ohjelmisto koostuu useammasta komponentista, joista tärkein ja merkittävin on tietysti itse tietokantamoottori, eli tietokantajärjestelmän ydin. Kuvassa 2.1 näkyy MS SQL Server 2005 -tuotteen komponentit, joista tässä IT-hallintajärjestelmässä on hyödynnetty erityisesti kolmea korostettua komponenttia: Database Engine (tietokantamoottori), Integration Services (tietojen siirrossa käytetty palvelu) ja Reporting Services (raporttien luonnissa käytettävä palvelu).



Kuva 2.1 *Microsoft SQL Server -tietokantapalvelimen komponentit [2].*

Kuvassa 2.1 olevat komponentit eivät suoranaisesti näy käyttäjälleen muuten kuin käytettävissä olevina palveluina. Palvelut ovat tietokantapalvelimella taustalla

suoritettavia prosesseja, joita käytetään varta vasten suunniteltujen rajapintojen kautta. Rajapintoja on olemassa erikseen muun muassa ylläpitoon ja tuotantokäyttöön. Ylläpitorajapintaa varten on olemassa SQL Server Management Studio -työkalu, jota käsitellään aliluvussa 2.3.

2.3. SQL Server Management Studio

SQL Server Management Studio on nimensä mukaisesti työkalu tietokantapalvelimen ylläpitoon. Sen avulla koko tietokannan asetuksia voidaan tarkastella ja muokata. Tämän työn kannalta tärkeämpää oli tietokantojen luominen ja niiden taulujen sisällön tutkiminen. Lisäksi SQL Server Management Studio mahdollistaa yksittäisten SQL-tietokantakomentojen suorittamisen. Kokonaisten tietokantaskriptien suorittaminen tapahtuu myös SQL Server Management Studion kautta, jolloin saadaan esimerkiksi luotua koko tietokanta alusta asti niin pitkälle, että se sisältää esimerkiksi kymmenittäin tauluja, useita näkymiä ja tallennettuja proseduureja, ja on täysin käyttöönotettavissa. Myös oikeusmääritykset voidaan tehdä tietokantaskriptien kautta vastaamaan lopullista käyttöönottotilannetta.

Tietokantaskriptien avulla tietokanta voidaan nopeasti luoda. Tietokantaskriptit helpottavat ja nopeuttavat testausta merkittävästi. Tulevaisuudessa tietokanta saatetaan siirtää toiselle palvelimelle esimerkiksi jonkin päivityksen yhteydessä, ja silloin uuden identtisen tietokannan luomiseksi riittää, että uudella palvelimella ajetaan tietokannanhallintaohjelmalla tietokantaskripti. Uuteen tietokantaan täytyy siirtää lisäksi vain vanhan tietokannan datat, ja koko tietokanta on siirretty. Tietokantaskripteillä voidaan teknisen määrityksen lisäksi tehdä myös dokumentointia, kun tietokannan objektien, kuten taulujen ja ominaisuuksien, nimeämiseen kiinnitetään erityistä huomiota.

2.4. Business Intelligence Development Studio

Business Intelligence Development Studio (BIDS) on Microsoftin kehittämä tuote, jolla käytetään muun muassa kuvassa 2.1 esiintyvien Integration Services (SSIS) - ja Reporting Services (SRSS) -komponettien palveluita. BIDS tarjoaa monipuolisen käyttöliittymän sekä SSIS -integraatioprojektien kehittämiseen että SRSS-raporttiprojektien luontiin ja testaukseen. Myös projektien debuggaus onnistuu BIDS:llä.

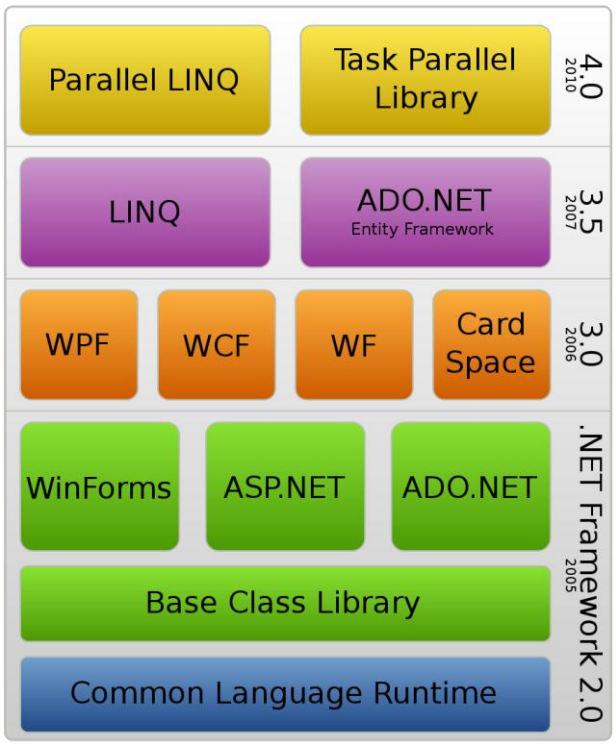
Microsoftin SQL Server Integration Services on SQL Server -tietokantapalvelimen komponentti, jolla voidaan helposti hoitaa esimerkiksi tietomassojen siirto järjestelmästä toiseen, ja samalla dataa voidaan manipuloida. SSIS-projektit luodaan BIDS-työkalulla ja niitä voidaan myös debugata sillä, jolloin nähdään mitä dataa missäkin välissä kulkee. Kun kehittäjä tietää, mitä dataa missäkin pitäisi olla, voidaan päästä käsiksi mahdollisiin ongelmiin.

2.5. Microsoft .NET Framework ja C#

Enemmän kuin työkalu, Microsoft .NET Framework on komponenttikirjasto, joka tukee jopa 20 eri ohjelmointikieltä, suosituimmat ovat C# ja VB.NET. C# on vuonna 2001 julkaistu .NET-komponettikirjastoa varten luotu ohjelmointikieli. ISO-standardin kieli on saanut vuonna 2003. Uusin versio .NET Frameworkistä on 4.0, joka on myös C#-kielen uusimman version versionumero.

Microsoft on kehittänyt C#-ohjelmointikielen .NET-komponenttikirjaston kehityksen yhteydessä. Kehittämisen päätavoitteina on Microsoftin mukaan ollut luoda useanlaisiin ympäristöihin soveltuva helppokäyttöinen, oliopohjainen ohjelmointikieli, jonka kansainvälistäminen olisi myös helppoa. Pohjana on ollut C-kielen syntaksi, Delphi- ja C++-ohjelmointikielien tehokkuus sekä esimerkiksi Javan tarjoama helppokäyttöisyys.

.NET Framework -komponenttikirjasto jakautuu kahteen pääosaan: luokkakirjastoon (Base Class Library) ja ajoympäristöön (Common Language Runtime, CLR). .NET-komponenttikirjasto sisältää useita eri luokkakirjastoja, mikä mahdollistaa kehittäjän keskittymisen olennaiseen, eli niin sanotun business-logiikan ohjelmointiin. Oheisessa kuvassa 2.2 on .NET-pinokaavio, jossa on esitetty alkeellinen aika- ja versiojana oikeassa reunassa.



Kuva 2.2 *.NET Framework -komponenttikirjaston pinokuvaaja [3].*

Kuvassa 2.2 pinokaavion alimpina ovat juuri CLR-ajoympäristö ja luokkakirjasto. Kaaviossa näkyvät myös WinForms- ja ADO.NET -komponentit, joista ensiksi mainittu sisältää graafisen käyttöliittymän toteutuksessa tarvittavia luokkia ja jälkimmäinen

puolestaan luokkia, joiden avulla saadaan otettua yhteys erilaisiin tietovarastoihin, muun muassa tietokantapalvelimiin.

2.6. Microsoft Visual C# 2010 Express

Microsoft Visual C# 2010 Express on ohjelmointiympäristö (IDE), joka on yksi kuudesta Microsoft Visual Studio 2010 Express -kokonaisuuteen kuuluvasta ohjelmasta. Microsoftin käytännön mukaan nimessä oleva Express-sana viittaa ilmaistyökaluun, mikä tässä tapauksessa tarkoittaa maksuttomuutta myös yrityskäytössä. Nimessä oleva C# puolestaan viittaa kyseiseen ohjelmointikieleen, joka on ainoa kieli, jota ohjelmointiympäristöllä on mahdollista ohjelmoida. Työkalulla pystytään luomaan Windows-sovelluksia, muun muassa konsoliohjelmia ja graafisen käyttöliittymän omaavia ohjelmia.

Maksuttomuus näkyy tietysti Visual C# Express -ohjelmointiympäristön ominaisuuksissa verrattuna maksulliseen Microsoft Visual Studio -työkaluun. Pienehköjen tässä työssä käsiteltävien sovellusten toteutus onnistuu hyvin myös ilmaistyökalulla.

2.7. Muut aputyökalut

Lisäksi työssä on hyödynnetty monia pienempiä työkaluja, joiden avulla on hoidettu yksinkertaisia rutiineja tehokkaasti. Eräs tällainen aputyökalu on Notepad++, joka on pieni, mutta erittäin monipuolinen tekstieditori.

3. TEORIAA

3.1. Perustietoa tietokannoista

3.1.1. Tietokantojen historiaa

Tietokantajärjestelmien historian kerrotaan alkaneen jo 1950-luvun loppupuolella, jolloin tiedot tallennettiin vielä reikäkorteille ja nauha-asemien nauhoille. Tällöin kehitettiin ohjelmointikieli, jossa tiedon käsittely oli eriytetty tietoa käyttävistä prosesseista. [4]

Tietojenkäsittelyn kehitys sai ison sysäyksen 1960-luvulla, kun levykkeet ja kiintolevyt mullistivat tietojen säilytyksen. Tietoja voitiin käsitellä entistä joustavammin ja niitä saatiin luettua kymmenissä millisekunneissa. Elettiin hierarkkisten tietokantojen ja verkkotietokantojen kulta-aikaa, jolloin tieto oli tallennettu listoiksi ja erilaisiksi puurakenteiksi. [5]

IBM:llä työskennellyt matemaatikko Edgar F. Codd julkaisi ensimmäisen kirjoituksensa relaatiotietomallista vuonna 1970, mitä voidaan pitää eräänlaisena käännekohtana tietokantojen historiassa. Kirjoituksessa esitellyt relaatiomallin keskeiset hyödyt olivat ilmeiset. Hyötyjä olivat muun muassa tiedon riippumattomuus käytettävissä olevasta tallennusvälineestä ja laitteistosta sekä korkean tason ei-proseduraalinen kieli datan käsittelyyn, toisin sanoen mahdollisuus automaattiseen korkean tason navigointiin tietokannassa. [4, 5]

Codd ei saanut kirjoitukselleen varauksetonta hyväksyntää heti julkistuksen jälkeen, kirjoitusta jopa pidettiin vain älyllisenä haasteena. IBM kehitti kuitenkin relaatiomallin mukaisen prototyypin 1970-luvun lopulla samaan aikaan useampien muidenkin kehittäessä vastaavaa prototyyppiä. IBM:n kehittämän prototyypin kyselykieli oli SQL, joka vakiintui muutamassa vuodessa standardiksi. Vaikka IBM olikin relaatiomallin kehityksessä vahva toimija, vuonna 1977 perustettu Oracle toi markkinoille ensimmäisen kaupallisen SQL-yhteensopivan tuotteen. [4]

Oliotietokantoja kehitettiin 1980-luvulla erittäin voimakkaasti, ja vuosikymmenen puolenvälin tietämissä ensimmäiset oliomallin mukaiset tietokannat olivat käyttövalmiina. Vuonna 1985 julkaistiin SQL-kielen standardista ensimmäinen versio. Julkaisustaan asti relaatiomallin mukaiset tietokannat ovat olleet käyttäjämäärän mukaan ylivertaisia muihin tietomalleihin verrattuna. [4, 5]

Aikajanaa karkeampi muoto jaotella eri tietomallit on jakaa ne eri sukupolviin. Oheinen taulukko 3.1 kokoaa vielä tietokantojen historian yhteen sukupolvittain eriteltynä.

Taulukko 3.1 *Tietokantojen historia sukupolvittain [6].*

Sukupolvi	Aika	Tietomalli	Esimerkkijärjestelmiä
Ensimmäinen	1960 - 1970	Tiedostopohjainen	VMS/VSAM
Toinen	1970-luku	Hierarkkinen tietomalli ja verkkotietomalli	IMS ADABAS IDS-II
Kolmas	1975 - nykyisyys	Relaatiomalli	DB2 Oracle MS SQL Server MySQL
Neljäs	1985 - nykyisyys	Oliomalli ja olio-relaatiomalli	Versant Objectivity/DB DB/2 UDB Oracle 11g
Tulevaisuus	nykyisyys - tulevaisuus	XML ja hybridi olio-relaatiomalli	dbXML Tamino DB2 UDB Oracle 11g MS SQL Server

3.1.2. Tietokantojen jaottelu

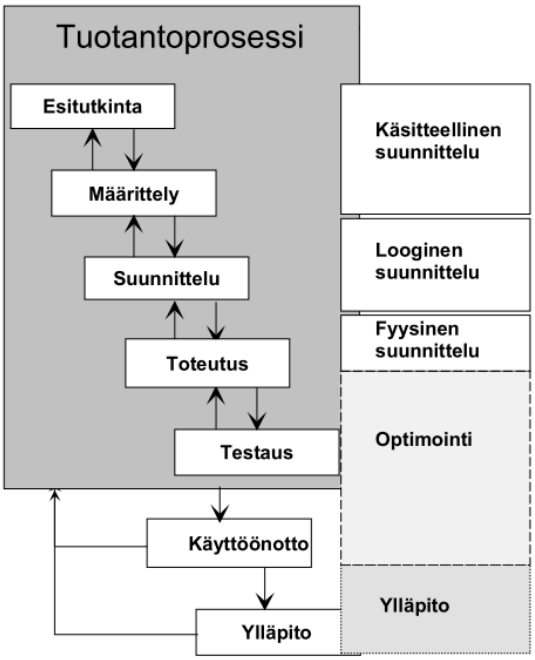
Tietokantojen vertailussa tietomallin mukainen jaottelu on ylivoimaisesti yleisin. Muita jaottelumuotoja ovat erilaiset määrälliset jaottelutavat: käyttäjien määrään perustuva tai tallennettavan tiedon määrään perustuvat. Käyttötarkoitus voi olla myös eräs peruste tietokantojen jaotteluun: on olemassa niin sanotusti yleiskäyttöisiä ja erikoiskäyttöisiä tietokantoja. [7]

Tietokantoja on mahdollista jaotella myös arkkitehtuurin perusteella. On olemassa esimerkiksi keskitettyjä ja hajautettuja tietokanta-arkkitehtuureja. Eräs jaottelutapa on kustannusperusteinen. Tietokantaratkaisuissa kalleimman kustannukset voivat helposti olla jopa tuhatkertaisia halvimpaan verrattuna. Kustannusperusteista jaottelua käytetään hyvin paljon. [7]

3.2. Tietokantasuunnittelun periaatteet

3.2.1. Suunnitteluprosessin käynnistys

Suunnitteluprosessin alkuvaiheessa on valittava tietomalli. Tämän työn tietomalliksi oli toimeksiannossa määritelty relaatiomalli. Oheinen kuva 3.1 esittää relaatiomallin suunnitteluprosessin vaiheittaisen edistymisen.



Kuva 3.1 *Relaatiomallin mukainen tietokannan suunnitteluprosessin vaiheittainen eteneminen [7].*

Kuvasta huomataan käsitteellisen suunnittelun käsittävän esitutkinta- ja määrittelyvaiheet. Looginen suunnittelu tapahtuu suunnitteluvaiheessa ja fyysinen suunnittelu toteutusvaiheessa. Testaus ja käyttöönotto ovat osa optimointia ja lopuksi on vielä ylläpito.

3.2.2. Käsitteellinen suunnittelu

Käsitteellisellä suunnittelulla tarkoitetaan sellaisen suunnitelman luomista, joka ei riipu tietokannanhallintajärjestelmästä eli tietomallista. Kuvasta 3.1 voidaan huomata, että käsitteellisen suunnittelun aikana suoritetaan esitutkintaa ja määrittelyä. Tässä suunnitteluvaiheessa kuvataan kohdealue sen omien käsitteiden, niiden ominaisuuksien ja suhteiden avulla. Käsitteellinen malli on määritelmän mukaan kohdealueen vääristymätön ja täydellinen esitys, jossa mahdolliset toteutusnäkökohdat jätetään huomioimatta. [7]

Käsitteelliseen malliin liittyy kiinteästi täydellisyys, riittävyys, semanttinen oikeellisuus, ymmärrettävyys ja minimaalisuus. Käsitteellisen mallin tavoite on luoda yhteinen kieli kuvaamaan kohdealuetta niin, että kaikki suunnitteluprosessissa mukana olevat henkilöt ymmärtävät samat käsitteet samalla tavalla. Käsitteellisen mallin voi luoda esimerkiksi UML-kielen luokkakaavionotaatiota käyttäen. [7]

3.2.3. Looginen suunnittelu

Looginen suunnittelu on vaihe, jossa käsitteellinen malli käännetään relaatioiksi, toisin sanoen relaatiomalleiksi. Tässä vaiheessa valitaan pääavaimet luokille: joko luonnolliset tai surrogaatit. Surrogaatilla tarkoitetaan varta vasten keksittyä pääavainta. Käännöksen yhteydessä suoritetaan normalisointi. [7]

Tietokannan normalisoinnilla tarkoitetaan vaiheittaista prosessia, jota noudattamalla pyritään saavuttamaan tietojen eheä tallennus ja tehokas saatavuus. Normalisoinnilla saadaan säästettyä tilaa, kun poistetaan tiedon redundanssi ja samalla yksinkertaistetaan päivityksiä, talletuksia ja ylläpitoa. Normalisoinnilla aiheutetaan kuitenkin hitautta hakuihin, sillä tieto on haettava useasta taulusta. [7]

Normalisointimuotoja on kehitetty tähän päivään mennessä jopa yhdeksän erilaista. Seuraavassa esitellään kuusi tunnetuinta normalisointimuotoa.

Ensimmäinen normaalimuoto (1NF) vaatii tietokannan kaikkien relaatioiden ominaisuuksien olevan atomisia, eli alkioina ei voi olla esimerkiksi tietoryhmiä tai toisia alkioita. Atomisuusehto saavutetaan tarvittaessa jakamalla relaatio osiin siten, että toistuvat ryhmät erotetaan omiksi tauluikseen. [5]

Toinen normaalimuoto (2NF) vaatii relaation olevan 1. normaalimuodossa ja jokaisen relaation ominaisuuden, joka ei ole avainominaisuus, olevan täydellisesti funktionaalisesti riippuva jokaisesta relaation avainehdokkaasta [7]. Jälleen normaalimuoto saavutetaan jakamalla relaatiot oikein omiksi tauluikseen [6].

Kolmas normaalimuoto (3NF) kieltää relaation ominaisuuksia riippumasta yhdestäkään ominaisuudesta, joka ei ole avainominaisuus. Sama asia voidaan siis sanoa niin, että kaikkien sarakkeiden tulee olla riippuvaisia vain pääavaimesta. Esimerkiksi postitoimipaikkaa ja postinumeroa ei voida tallentaa samaan relaatioon, jollei niistä toinen ole pääavain. Kolmanteen normaalimuotoon päästään jakamalla relaatiot tarvittavan moneksi tauluksi. [7]

Boyce-Codd -normaalimuoto (BCNF) saavutetaan, kun relaatio ei sisällä tarpeetonta toisteista tietoa. Boyce-Codd -normaalimuoto on oikeastaan vain tiukempi versio kolmannesta normaalimuodosta. [6]

Neljäs normaalimuoto (4NF) saavutetaan, kun relaatio on Boyce-Codd -normaalimuodossa, ja kun relaatiossa ei ole ei-triviaaleja riippuvuuksia. Relaatio on muutettavissa neljänteen normaalimuotoon jakamalla se useampaan tauluun. [7]

Viides normaalimuoto (5NF) sanoo, että ositetuista relaatioista pitää saada takaisin alkuperäiset relaatiot käyttämällä luonnollisia liitoksia. Joissakin tapauksissa alkuperäinen relaatio pitää osittaa kerralla useampaan osaan. [7]

Normalisoinnin vastakohtana on denormalisointi, jolla saadaan suorituskykyä yhdistämällä tietoja takaisin yhdeksi tauluksi. Denormalisoinnin nopeuttamisen haittapuolena on muutoksien ja päivityksien hidastuminen. [7]

3.2.4. Fyysinen suunnittelu

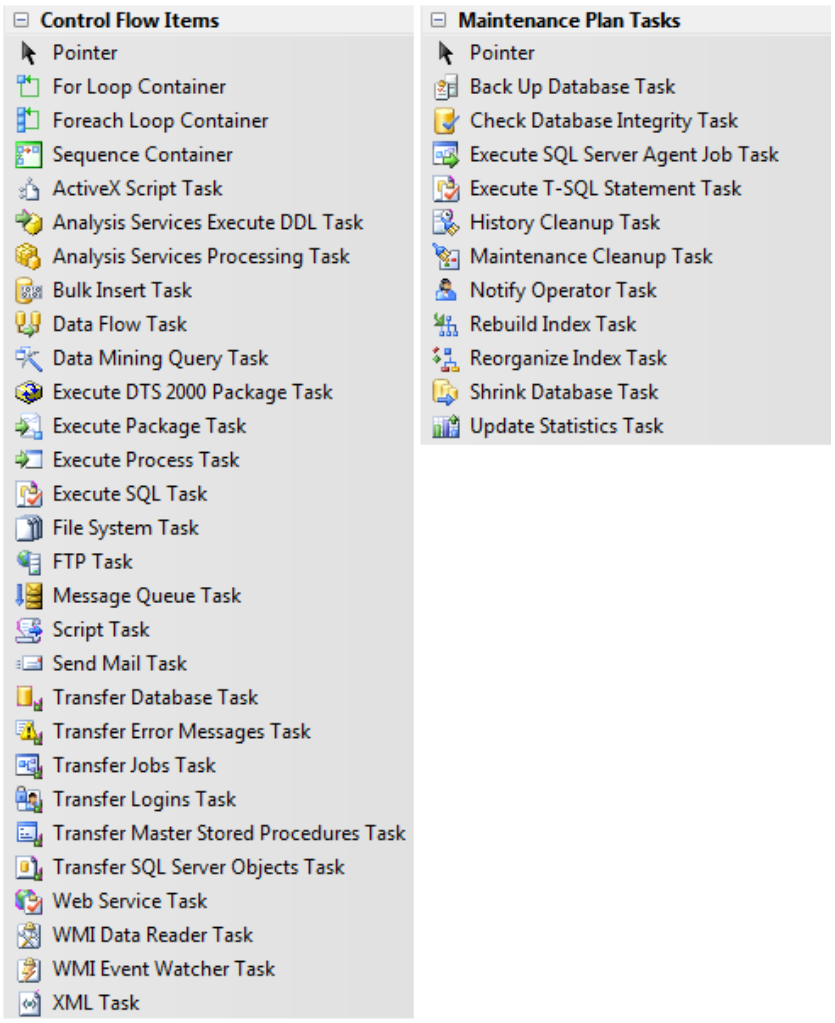
Fyysisellä suunnittelulla tarkoitetaan vaihetta, jossa tietokannasta luodaan tietokannanhallintajärjestelmästä riippuvainen kuvaus eli toteutus tietylle tietokannanhallintajärjestelmälle. Tässä vaiheessa valitaan tiedostorakenne sekä suunnitellaan indeksit, käyttäjien näkymät ja turvallisuusmekanismit. Tietokannan suorituskykyvaatimuksiin pyritään pureutumaan indeksien suunnittelulla niin, että lopullinen järjestelmä on suorituskyvynkin kannalta käytettävä. [7]

Lopputuloksena tästä suunnitteluvaiheesta saadaan muun muassa joukko taulujen luontikomentoja ja muita määrityskomentoja. Suorittamalla ne saadaan luotua itse tietokanta. [7]

3.3. Integraatioprojektin periaatteet

Microsoftin SQL Server Integration Service (SSIS) -projektien teoriaa on hyvin vähän käsitelty kirjallisuudessa ja julkaisuissa. Tämä johtuu tuotteen nuorehkosta iästä ja luultavasti myös siitä, että sen käyttäminen on alkeellisimmillaan hyvin suoraviivaista.

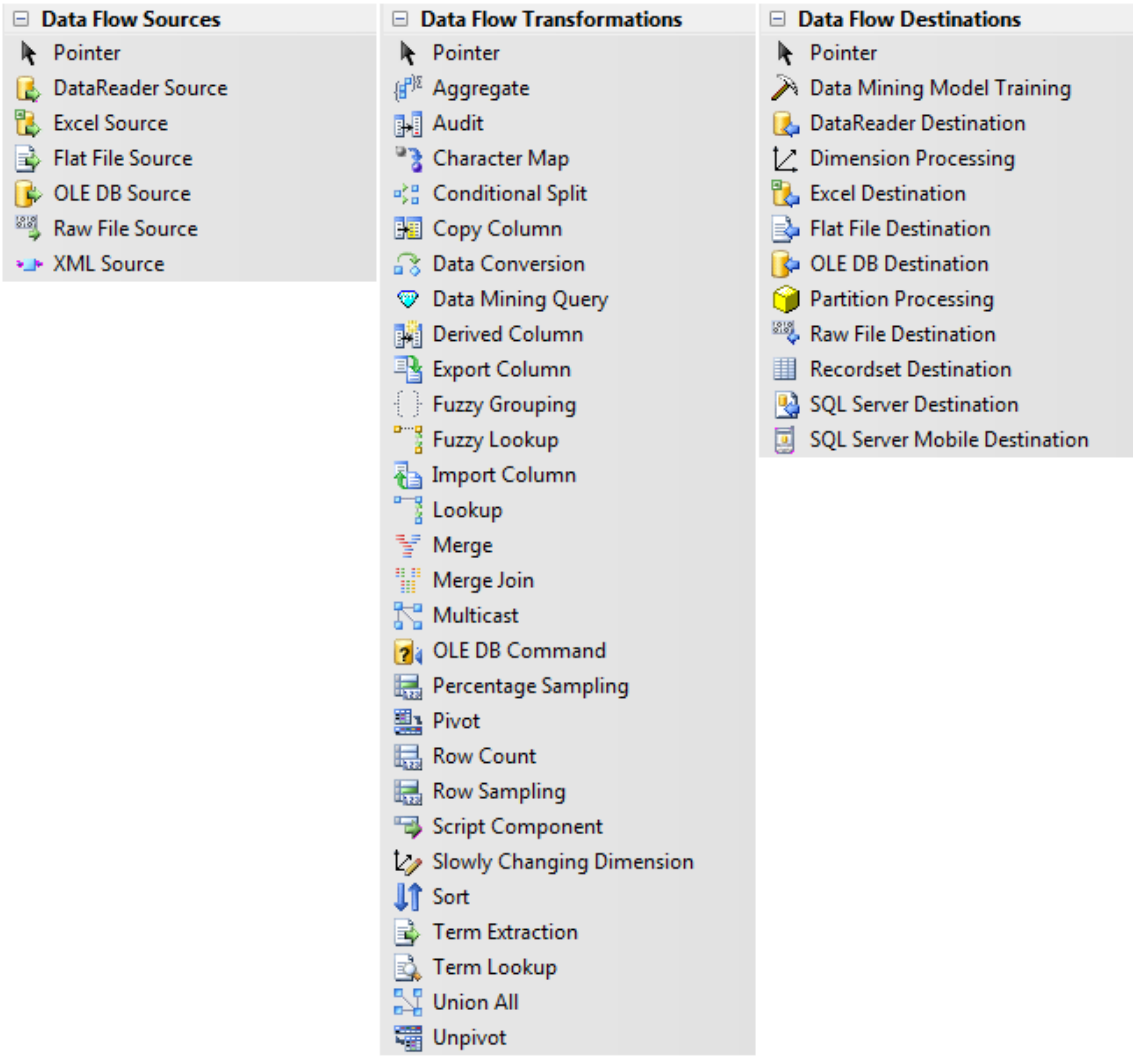
SSIS-projekti koostuu niin sanotusta prosessikuvaksesta (engl. Process Flow), joka sisältää erilaisia tehtäviä (engl. Task). Prosessikuvaus määrittelee tehtävien lukumäärän ja niiden suoritusjärjestyksen. Tehtävien lukumäärää ei ole rajoitettu. Suoritusjärjestys voi olla ehdollinen, mistä on hyötyä etenkin säännöllisissä automaattiajoissa. [8] Kuvassa 3.2 on kuvaruutukaappauksena esitetty tarjolla olevat tehtävät, jotka on jaettu kahteen kategoriaan.



Kuva 3.2 *Kuvakaappaus tarjolla olevista prosessikuvauksen tehtävistä.*

Erilaisten tehtävien määrä on valtava. Yksinkertaisten SSIS-projektien käytetyimpiä tehtäviä ovat Data Flow ja Execute SQL Task. Execute SQL Task -tehtävä on nimensä

mukaan SQL-lauseen ajamiseen tarkoitettu tehtävä. Data Flow -tehtävät määritellään erikseen, ja niissä datan voidaan ajatella kulkevan eräänlaisessa putkessa vaiheesta toiseen. Data Flow -tehtävissä voi olla kolmen tyyppisiä komponentteja: datan lähteitä, kohteita ja transformaatioita eli tietomuunnoskomponentteja. [8] Nämä Data Flow:n eri komponentit on listattu kuvan 3.3 kuvaruutukaappaukseen.



Kuva 3.3 *Kuvakaappaus Data Flow'n komponenttivalikoimasta.*

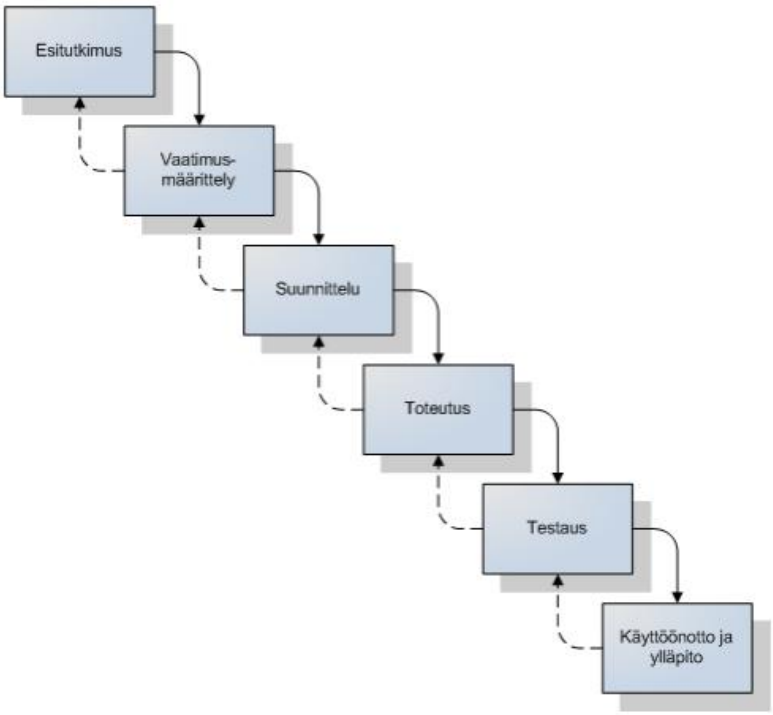
Datan lähteet ja kohteet voivat olla esimerkiksi tietokantoja tai pelkkiä tietokonetiedostoja. Tiedon muokkausta varten on useita transformaatioita, joilla voidaan ratkaista laaja joukko erilaisia ongelmia. Ongelmatapauksista visaisimmat ratkennevat viimeistään Script Component -transformaatiolla, jolla voidaan käytännössä luoda kulloisenkin ongelman ratkaiseva transformaatio. Script Component -transformaation monimutkaisuutta rajoittaa vain luojan mielikuvitus, ja se voidaan kirjoittaa C#- tai VB.NET-ohjelmointikielillä. [8]

3.4. Ohjelmistotuotannon vaihejakomallit

3.4.1. Vesiputousmalli

Vuonna 1970 Winston Royce esitteli prosessimallin, joka myöhemmin nimettiin vesiputousmalli. Roycen malli oli viallinen ja käytäntöön soveltumaton, mutta silti se omaksuttiin laajalti ja sitä käytettiin yleisenä prosessimallina 1980- ja 1990-luvuilla. [9]

Vesiputousmalli jaetaan lähteestä riippuen viidestä seitsemään vaiheeseen. Kuvassa 3.4 on esitelty vesiputousmallin vaiheet: ehyt nuoli kuvaa etenemistä ja katkoviiva puolestaan paluuta edelliseen vaiheeseen.



Kuva 3.4 *Vesiputousmallin vaiheet suoritusjärjestyksessä ylhäältä alas.*

Perinteisen ajattelutavan mukaan vesiputousmallissa ei saa edetä seuraavaan vaiheeseen, ennen kuin edellinen vaihe on täysin valmis. Paluu on mahdollista siinä tilanteessa, että takaisin paluulla korjataan seuraavissa vaiheissa havaittuja virheitä. [10]

Vesiputousmallista on monia variaatioita. Yksi niistä on niin sanottu inkrementaalinen kehitys, joka yhdistää useita vesiputousmalleja peräkkäisiksi. Tällöin kehitys etenee pikku hiljaa vesiputousmallista toiseen. [9]

3.4.2. Ketterät menetelmät

Vesiputousmallia pidetään jäykkänä ja sen orjallinen noudattaminen on harvinaista. Tämän vuoksi ketterät menetelmät ovat vallanneet jalansijaa ohjelmistotuotannon vaihejakomallikentässä.

Vuonna 2011 useat ketterien kehitysmenetelmien puolestapuhujat julkaisivat yhteisen tapaamisen päätteeksi ketterän manifestin, jota pidetään ketterän kehityksen perusmääritelmänä. Taulukossa 3.2 on esitetty ketterä manifesti.

Taulukko 3.2 *Ketterän kehityksen manifesti [9].*

Yksilöt ja vuorovaikutus ovat tärkeämpiä kuin prosessit ja työkalut
Toimiva ohjelmisto on tärkeämpi kuin kattava dokumentaatio
Yhteistyö ja kumppanuus ovat tärkeämpiä kuin sopimusneuvottelut
Muutoksiin reagointi on tärkeämpää kuin suunnitelman noudattaminen

Manifestissa vasemmalla tummennettuna olevat asiat ovat oikealla olevia tärkeämpiä ja niitä korostetaan, mutta oikealla puolella olevat eivät ole merkityksettömiä eikä niitäkään saa jättää huomiotta. [9]

Tunnetuimpia ketteriä kehitysmenetelmiä ovat Scrum ja XP (eXtreme Programming), joista Scrumia pidetään nykyään suosituimpana ketteränä ohjelmistokehityksen menetelmänä. Scrum pyrkii yksinkertaisuudellaan ja suoraviivaisuudellaan tarjoamaan mahdollisuuden keskittyä tärkeimpään eli työn tavoitteisiin. Scrumissa pyritään ylläpitämään tiivistä asiakkaan ja kehittäjien välistä kommunikaatiota, ja projektin mahdollisiin vaatimusmuutoksiin pyritään reagoimaan nopeasti. [9]

3.5. Ohjelmiston elinkaaren vaiheet

3.5.1. Esitutkimus

Ohjelmistoilla on lähes aina samat elinkaaren vaiheet riippumatta valitusta vaihejakomallista. Elinkaaren ensimmäinen vaihe on usein esitutkimus, jossa selvitetään toteutettavan järjestelmän yleiset vaatimukset. [11] Se vastaa kysymykseen, miksi järjestelmä tai ohjelmisto tehdään, tai miksi ei. Esitutkimusvaiheeseen liittyy haasteita, joista ylivoimaisesti merkittävin on asiakkaiden todellisten tarpeiden selvittäminen ja tulkitseminen. Alussa tehdyt virheet ja puutteet kertautuvat suunnittelun edetessä. [10]

3.5.2. Vaatimusmäärittely

Vaatimusmäärittelyvaiheessa selvitetään tarkasti eri sidosryhmien tarpeet järjestelmää kohtaan, mutta siinä ei oteta vielä kantaa siihen, miten ne aiotaan toteuttaa. Syntyvää dokumenttia kutsutaan toiminnalliseksi määritelmäksi. [12]

Vaatimukset jaetaan perinteisesti kahteen pääkategoriaan, toiminnallisiin ja ei-toiminnallisiin. Toiminnalliset vaatimukset kertovat, mitä järjestelmän odotetaan tekevän. Ei-toiminnalliset puolestaan ottavat kantaa siihen, miten toiminnalliset vaatimukset toteutetaan, ne ovat siis niin sanotusti laadullisia vaatimuksia. Laadullisia

vaatimuksia ovat muun muassa oikeellisuus, suorituskyky, käytettävyys, turvallisuus ja ylläpidettävyys. [12]

3.5.3. Ohjelmistosuunnittelu

Ohjelmistosuunnittelu on vaihe, jossa toiminnallinen määrittely muutetaan tekniseksi määrittelyksi siitä, miten järjestelmä on tarkoitus toteuttaa. Suunnitteluvaihe voidaan jakaa kahteen osaan: arkkitehtuurisuunnitteluun ja moduulisuunnitteluun. Arkkitehtuurisuunnittelu on korkeamman tason suunnitelma, jossa ohjelma on jaettu pienempiin kokonaisuuksiin, moduuleihin. Moduulisuunnittelu tarkoittaa kunkin yksittäisen moduulin suunnittelua. [12]

Arkkitehtuurisuunnittelun tavoitteena on selkeyttää isoja kokonaisuuksia jakamalla ne pienemmiksi. Ohjelman arkkitehtuuri vaikuttaa erityisesti tehokkuuteen, robustisuuteen, jaoteltavuuteen ja ylläpidettävyyteen. Arkkitehtuurisuunnittelussa voidaan käyttää hyväksi suunnittelumalleja, joita on olemassa useita erilaisia. [13]

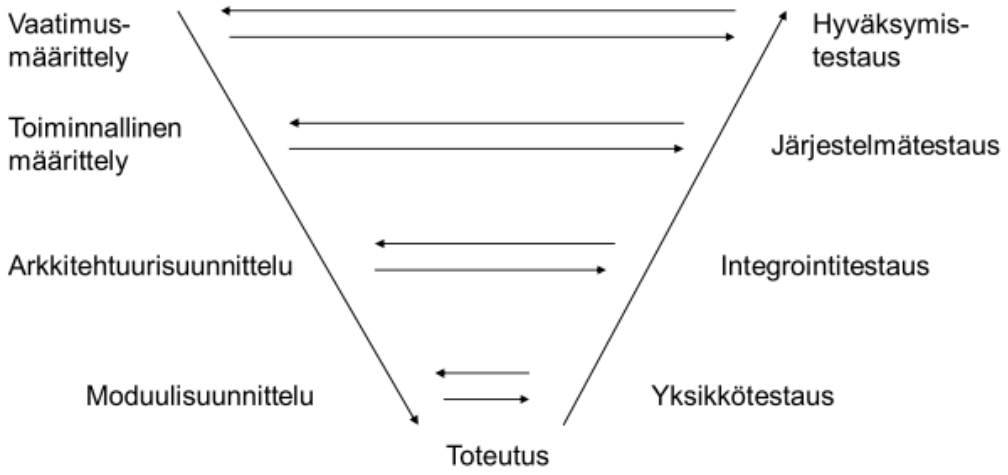
Moduulisuunnittelussa pureudutaan yhteen pienempään osakokonaisuuteen, jota kehitetään muista erillään. Moduulista voidaan luoda uudelleenkäytettävä tehokas osakokonaisuus, kun otetaan kaikki muut moduulin palveluita käyttävät osat huomioon. [12]

3.5.4. Toteutus

Toteutusvaiheessa kirjoitetaan varsinainen ohjelmakoodi, joka toteuttaa moduulit ja koko järjestelmän moduulit yhdistävällä toiminnallisuudella. Mikäli edelliset vaiheet on tehty kurinalaisen täsmällisesti, toteutus sujuu nopeasti ja suoraviivaisesti. [12]

3.5.5. Testaus

Testaus jaetaan perinteisesti neljään osaan: yksikkötestaukseen, integrointitestaukseen, järjestelmätestaukseen ja hyväksymistestaukseen. Puhutaan niin sanotusta testauksen V-mallista, joka on esitetty kuvassa 3.5. Vaakasuorassa oikealle osoittava nuoli kuvaa testauksen suunnittelua ja vasemmalle osoittava nuoli tulosten variointia, testituloksia peilataan aina määriteltyihin ominaisuuksiin. [14]



Kuva 3.5 *Testauksen V-malli [14].*

Moduulien testaus suoritetaan limittäin toteutuksen kanssa. Vaihetta kutsutaan yksikkötestaukseksi. Kutakin moduulia varten on tarkoituksenmukaista luoda testipenkki, joka ottaa huomioon kaikki testitapaukset. [14]

Integrointitestaus tehdään, kun moduulit on yhdistetty suuremmiksi kokonaisuuksiksi. Integrointitestauksessa testataan moduulien yhteistoimintaa ja niiden välisiä rajapintoja. [14]

Järjestelmätestaus suoritetaan integrointitestauksien jälkeen, nimensä mukaisesti järjestelmätestauksessa testataan koko järjestelmää. Ajallisesti järjestelmätestaus vie eniten aikaa, ja vasta täällä vaiheessa löydetyt virheet tulevat yleensä hyvin kalliiksi. [14]

Hyväksymistestaus on viimeinen testausvaihe, jossa pyritään selvittämään, onko tuote sopimusten mukainen. Hyväksymistestaus perustuu asiakasvaatimuksiin ja siinä testauskohteena on valmis tuote. [14]

3.5.6. Käyttöönotto

Käyttöönottovaiheeseen liittyy projektista riippuen muun muassa ohjelman asennus, vanhojen tietojen konvertointi ja käyttäjien ohjeistaminen uuteen ohjelmaan. Suurissa projekteissa luodaan käyttöönottosuunnitelma, jotta käyttöönotto sujuisi ongelmitta.

3.5.7. Ylläpito

Ylläpitovaihe ei varsinaisesti kuulu enää ohjelmiston suunnitteluun, mutta muutoin se on tärkeässä roolissa ohjelmistohankkeissa esimerkiksi kustannusten osalta. Ylläpitoon ja sen helppouteen vaikuttavat suuresti aiemmat ohjelmistosuunnitteluprojektin vaiheet ja niiden dokumentointi.

4. LÄHTÖTIEDOT

4.1. Vanha kirjanpitojärjestelmä

Entinen kaluston- ja lisenssienhallintajärjestelmä on peräisin ajalta, jolloin yrityksen koko oli pieni, noin kymmenen henkeä. Silloin myös laitteita oli vähän, ja ne olivat helposti hallittavissa jopa yhdellä Excel-taulukkolaskentatiedostolla. Samoin lisenssejä käytettiin vähemmän ja eri ohjelmia oli vain kourallinen - kaikki lisenssit saattoivat olla merkittynä yhteen Excel-tiedostoon.

Sittemmin työntekijöiden määrä on kasvanut ja samalla luonnollisesti myös kaluston määrä on kasvanut moninkertaiseksi. Alkuaikojen kymmenen henkilön yrityksestä on kasvanut muun muassa yritysostojen kautta useamman sadan henkilön suurehko PK-yritys. Nykyisin yrityksellä on käytössään kymmenittäin eri ohjelmistoja, minkä vuoksi jokaisella ohjelmistolla on oma Excel-tiedosto lisenssikirjanpitoa varten.

Yrityksen kuluja seurataan nykyään todella tarkasti erilaisten raporttien avulla, viime aikoina trendinä on ollut ottaa mukaan myös IT-kaluston ja lisenssien kulujen raportointi. Entisen kirjanpitojärjestelmän Excel-tiedostot eivät tue raporttien helppoa luomista. Juuri tästä syystä Excel-tiedostot on aikanaan viety Access-tietokantaan. Tuossa yhteydessä tietokantojen suunnitteluun ei uhrattu erityisesti aikaa, vaan Excel-taulukko vain vietiin sellaisenaan muutamaan tietokannan tauluun.

Kalustokirjanpidossa on oleellista paitsi tietää, kenellä yrityksen nykyisin omistamat laitteet ovat käytössä, myös säilyttää vanhojen, jopa jo hävitettyjen, laitteiden historiatietoa muun muassa arvokkaiden kommenttien vuoksi. Nykyisen Access-tietokannan sisällä on neljä taulua, jotka on esitelty taulukossa 4.1.

Taulukko 4.1 *Vanhan Access-tietokantakirjanpidon taulut ja niiden selitykset.*

Taulun nimi	Selitys
Laitetyypit	Osa käytetyistä laitetyypeistä oli listattuna tähän tauluun itse generoidun järjestysnumeron ja ryhmänimen kanssa. Tällä listauksella saatiin joihinkin raportteihin haettavat laitteet ryhmiteltynä tiettyyn järjestykseen: esimerkiksi keskusyksiköille looginen paikka oli ennen hiiriä ja näppäimistöjä. Listassa oli vain niitä laitetyyppejä, joita raporteissa tarvittiin.

Taulun nimi	Selitys
Laitteet_uusi	Tässä taulussa pidettiin nykyisin vielä aktiivisena olevia laitteita. Laitteiden tiedot syötettiin tänne niiden saapuessa yritykseen. Myöhemmin laitteiden käyttäjätietoa päivitettiin tarpeen mukaan. Tämän päivityksen pystyi tekemään vain osa IT-osaston henkilöstöstä, jotta laitekirjanpito pysyi jotenkin ehjänä.
Poistuneet laitteet	Yrityksen käytöstä tai omistuksesta poistuneet laitteet ajettiin tähän tauluun Laitteet_uusi -taulusta käytöstä poistamisensa jälkeen. Tällä tavalla vanhoista laitteista säästyi edes jotakin historiatietoa. Tämän taulun kentät ovat siis identtiset Laitteet_uusi -taulun kanssa, jotta tietojen siirtäminen oli ylipäänsä mahdollista.
SampoKohdenumerot	Leasing-laitteiden tiedoissa on Laitteet_uusi -taulussa kerrottu laskun numero, joka linkittyy tämän taulun Laskun nro -kenttään. Tätä kautta selviää leasing-kohteen numero ja laitteen toimittanut yritys.

Näissä kahdessa laitteita sisältävässä taulussa on rivejä yhteensä noin 3000. Alkuajan muutaman kymmenen laitteen Excel-taulusta on kasvanut ajan myötä massiivinen määrä dataa.

4.2. Järjestelmäkokonaisuuden osa-alueet

4.2.1. Kalusto

Moninainen joukko IT-laitteita muodostaa kaluston. Niihin kuuluvat muun muassa tietokoneiden keskusyksiköt, näppäimistöt, hiiret ja näytöt, nykyaikana myös esimerkiksi kannettavien tietokoneiden telakat. Kaluston kirjanpidossa oleellisinta on heti laitteen saavuttua ottaa se mukaan kirjanpitoon. Toinen kriittinen kohta laitteen elinkaaressa on sen hävittäminen. Laitteen saapumisen yhteydessä siitä tehdään kirjanpitomerkintä ennen varsinaista käyttöönottoa. On hyvin tärkeää kirjata laitteet heti kirjanpitoon, jotta voidaan olla varmoja, että kaikki laitteet sinne päätyvät.

Laitteen saapuessa siitä otetaan ylös tunnistetiedot kuten valmistaja, tarkka malli ja laitteen yksilöivä tuote ID -tunniste, niin sanottu PID (engl. Product ID). Tämän lisäksi kullekin laitteelle annetaan yrityksen sisäinen kalustonumero, jolla laitteet on helpompi yksilöidä. Näin saadaan koko laitekannan laajuinen samanlainen numerointitapa. Laitenumerot merkitään laitteisiin tarratulostimella tehdyillä tarroilla ja kirjataan kirjanpitoon laitteen tietoihin.

Uusimman yrityksessä käyttöön otetun käytännön mukaan tietokonepaketit merkitään yhden nelinumeroisen kalustonumeron alle niin sanotulla pistemerkinnällä

(esim. 1234.1). Tällä käytännöllä saadaan selkeytettyä kalustokirjanpitoa ja säästetään numeroita.

Lähtötietoina saadun vanhan kirjanpitojärjestelmän datasta voitiin havaita laitteiden kirjon olevan erittäin laaja. Oheisessa taulukossa 4.2 on pääosa käytössä olevien laitteiden tyypeistä.

Taulukko 4.2 *Käytössä olevien laitteiden tyyppikirjoa aakkosjärjestyksessä lueteltuna.*

3D-ohjain	KVM-kytkin	palvelinjärjestelmä	ulk. diskettiasema
Centronics-jakaja	kytkin	paperituhooja	ulk. USB verkkokortti
dataprojektori	laskukone	piirturi	ulk. USB-levy
digitaalikamera	leikkuri	skanneri	ups
hiiri	levykeasema	tarrakirjoitin	USB-hubi
kannettava	nauha-asema	telakka	vastaanotin
keskusyksikkö	näppäimistö	tulostin	web-kamera
kirjoituskone	näyttö	tulostinjakaja	Wlan-laite
kopiokone	palomuuri	ulk. cd-asema	

Laitteiden suuri kirjo tuo tietokannan suunnitteluun haasteita, sillä kaikkien laitteiden tiedot pitää voida kirjata samaan järjestelmään mahdollisimman vähin tauluin, mutta kuitenkin täydellisesti. Tietokannan suunnittelussa pitäisi lisäksi pyrkiä välttämään toisteisuutta, mutta tietokantasuunnitelman tulisi silti olla yksinkertainen.

4.2.2. Lisenssit

Ohjelmiston käyttöoikeutta kuvaa lisenssi. Lisenssit voidaan jakaa kahteen päätyyppiin, kiinteisiin lisensseihin ja verkkolisensseihin. Verkkolisenssejä kutsutaan joskus myös nimellä kelluva lisenssi, mikä juontaa juurensa niiden käytön joustavuudesta. Lisenssi on lähes aina jokin konkreettinen asia, esimerkiksi numerosarja, tietokonetiedosto, fyysinen tietokoneeseen liitettävä laite tai jokin edellä mainittujen yhdistelmä. Fyysinen laite voi liittyä sekä kiinteisiin lisensseihin että verkkolisensseihin. Laite saatetaan liittää esimerkiksi USB-portin välityksellä tietokoneeseen: kiinteiden lisenssien tapauksessa työasemaan, verkkolisensseissä lisenssipalvelimena toimivaan palvelintietokoneeseen.

Pelkkä numerosarja voi jo itsessään olla lisenssi, tai sitä voidaan käyttää aktivointivälineenä, jolla lisenssi saadaan otettua käyttöön esimerkiksi Internetin välityksellä. Molemmat tavat pätevät sekä kiinteisiin lisensseihin että verkkolisensseihin.

Verkkolisenssit ovat useimmissa tapauksissa kiinteitä lisenssejä joustavampi vaihtoehto, minkä vuoksi niitä pidetään etenkin IT-osaston kannalta helpommin ylläpidettävinä. Käytännössä verkkolisenssejä varten on olemassa yrityksen sisäinen lisenssipalvelin, jolta ohjelmaa käynnistäessä varataan yksi lisenssi käyttöön. Verkkolisenssin käytön edellytyksenä on verkkoyhteys yrityksen sisäiseen verkkoon ja kyseistä lisenssiä tarjoavaan lisenssipalvelimeen.

Lisenssipalvelimella voi olla tietyn ohjelman lisenssejä käytettävissä esimerkiksi 20 kappaletta samalla, kun kyseinen ohjelma on asennettuna yli 50 tietokoneeseen. Erinäisten aputyökaluohjelmien kanssa edellä mainittu joustavuus on ensiarvoisen tärkeää, etenkin silloin kun ohjelman käyttäminen on satunnaista. Edellä mainitulla järjestelyllä ei suljeta pois jatkuvaakaan käyttöä, sillä lähes kaikissa verkkolisenssijärjestelmissä on mahdollista lainata lisenssin käyttöoikeus pidemmäksi aikaa. Lisenssi pysyy varattuna, vaikka verkkoyhteyttä lisenssipalvelimeen ei olisikaan. Tällöin saadaan käytettyä verkkolisenssejä kiinteän lisenssin tavoin esimerkiksi matkoilla ollessa.

Lisenssien taloudelliseen puoleen liittyvät olennaisesti käsitteet osto ja ylläpito. Ostolla tarkoitetaan sitä hetkellistä tapahtumaa, jolloin jonkin tietyn ohjelman käyttöoikeus eli lisenssi hankitaan. Usein ostettua ohjelman versiota saadaan käyttää niin pitkään kuin se toimii. Ylläpidolla tarkoitetaan puolestaan vuotuista maksua ohjelmavalmistajalle. Ylläpitomaksu mahdollistaa ohjelman joustavan päivittämisen uusimpaan versioon sekä antaa yritykselle oikeuden tukipalveluiden vapaaseen käyttämiseen. Ostomaksu on siis kertaluonteinen erä, kun taas ylläpitomaksu on jatkuva, vuosittainen kuluerä.

4.2.3. Laskut ja leasing-kohteet

Laskutus liittyy kiinteästi koko yrityksen toimintaan, myös IT-laitteistoihin ja lisensseihin. Useimmat työasemat ovat nykyään yrityksen käytössä leasing-sopimuksella. Tällöin yritys ei itse omista laitteita, yrityksellä on niihin vain käyttöoikeus leasing-sopimuksen määrittelemän ajanjakson ajan. Leasing-palvelua käyttävän yrityksen on itse pidettävä huolta siitä, että laitteet palautetaan ajallaan. Tämän vuoksi IT-laitteiden kirjanpitoon on tärkeää merkitä kunkin laitteen leasing-sopimuksen umpeutumisajankohta.

Rahoitusyhtiö toimii leasing-palvelua tarjoavana yrityksenä, kun taas laitteiden toimittajana on usein jokin toinen yritys. Yleensä rahoitusyhtiö kerää yhteen kaikkien saman leasing-jakson laitteiden laskut niin sanotusti saman leasing-kohteen alle. Yhden leasing-kohteen kaikkien laitteiden vuokra-aika päättyy samana päivänä, ellei joitakin laitteita oteta niin sanotusti jatkolle. Jatkolle otetut laitteet säilyvät edelleen saman leasing-kohteen alla, mutta leasing-kohteen päättymispäivä päivitetään uuden jatkosopimuksen mukaiseksi.

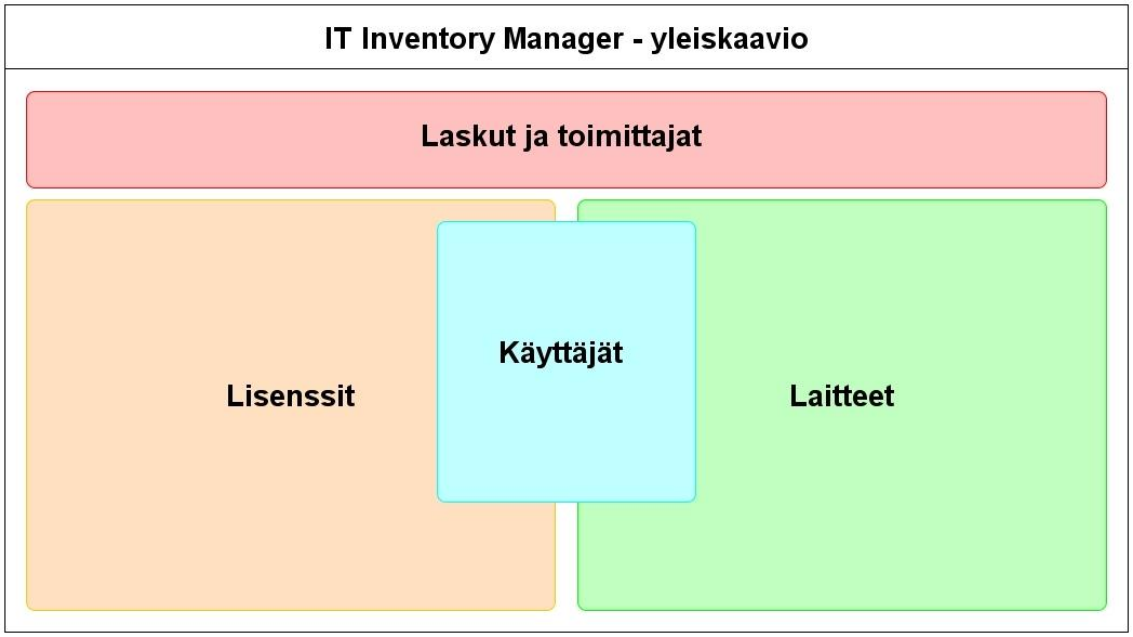
Toimittajien yhteystietojen kirjanpito on tärkeää ja luonnollisimmin se onnistuu laskujen kautta, kullakin laskulla on toimittaja ja toimittajalla vähintään yksi yhteyshenkilö. Sama ketju pätee sekä laitteisiin että lisensseihin. Erityisesti IT-henkilöiden lomien aikana olisi ensiarvoisen tärkeää, että yllättävien yhteydenottotarpeiden sattuessa olisi jokin tieto siitä, mikä taho on toimittanut laitteen tai lisenssin. Tätä kautta voidaan ottaa yhteyttä toimittajatahoon ja jouhevasti saada hoidettua muun muassa takuuasiat.

Lisenssien laskutuksessa on huomioitava sekä ostolaskut ja ylläpitolaskut, ne ovat käsitteellisesti eri asioita ja kirjanpitojärjestelmän tulee kohdella niitä myös eri tavoin.

5. VAATIMUKSET

5.1. Tietokanta

Työn aloitusvaiheessa oli tiedossa toimeksianto ja luvun 4 lähtökohdat, ja niiden pohjalta lähdettiin selvittämään työn vaatimuksia. Tietokantaan piti pystyä toimeksiannon mukaan tallentamaan käsitteellisesti kaikki se tieto, joka oli tallennettu myös vanhaan Access-tietokantaan. Tietokannan tuli muodostua kuvassa 5.1 näkyvistä pääosa-alueista, joita ovat lisenssit, laitteet, käyttäjät sekä laskut ja toimittajat.



Kuva 5.1 *Vaatimuksena määritellyt tietokannan osa-alueet.*

Käyttäjätiedot saadaan erillisestä ulkopuolisesta järjestelmästä, joten käyttäjistä on säilytettävä vain aivan välttämättömin tieto. Muu tarvittava tieto käyttäjistä saadaan automaattisesti päivittyvästä täydellisestä lähteestä.

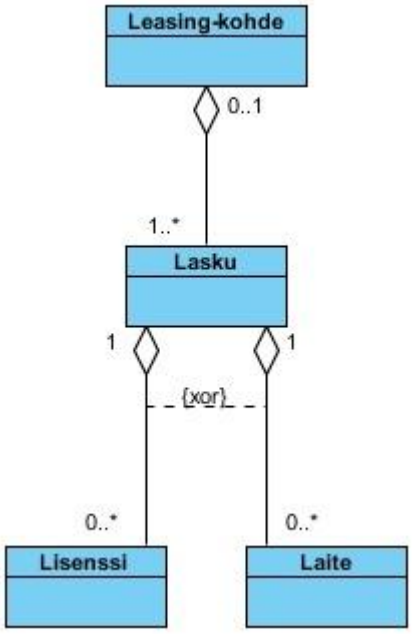
Kuvan 5.1 osa-alueiden piti liittyä toisiinsa niin kuin ne luonnollisestikin liittyvät. Yhdellä lisenssillä on lasku, käyttäjä ja mahdollisesti myös laite, johon lisenssi on asennettu. Lisenssiin kuuluvan laskun kautta pystytään navigoimaan lisenssin toimittajaan. Lisenssillä voi ostolaskujen lisäksi olla ylläpitolaskuja. Laitteella puolestaan pitää olla käyttäjä tai jokin muu hyväksytty tilatieto (status), lasku. Laitteen toimittajaan pitää voida navigoida laskun kautta. Laitteeseen pitää voida myös liittyä laskun kautta leasing-kohde, joka määrittelee laitteen käyttöoikeusajan.

Luotavan tietokantarakenteen tulee olla selkeä ja mahdollisimman yksinkertainen mutta silti riittävä. Lisäksi tarpeetonta toisteisuutta pitää välttää. Tietokantaratkaisun tulee noudattaa hyvien suunnittelutapojen tietokantamalleja.

Tietokantaratkaisun suunnittelussa oli otettava huomioon, ettei siitä ollut tarkoitus poistaa vaan ainoastaan lisätä siihen tietoa. Lähinnä se tarkoittaisi aikaleimojen lisäämistä tietoihin, esimerkiksi laitteilla olevien tilatietojen mukana piti tallettaa tieto siitä, milloin kyseinen tilatieto oli ollut voimassa. Tällä tavalla järjestelmään jäisi tieto esimerkiksi laitteiden ja lisenssien käyttöhistoriasta, mikä oli yksi vanhan järjestelmän merkittävimmistä puutteista.

Kirjanpito-objektien (laitteet, lisenssit, laskut, jne.) kommentointimahdollisuuteen haluttiin kiinnittää huomiota. Pitäisi esimerkiksi olla mahdollista tallentaa laitteen yhteyteen kommentti siitä, jos sen toiminnassa oli huomattu jokin satunnainen vika. Tällaisilla tiedoilla voi olla kullanarvoinen merkitys myöhemmin, kun voidaan kirjanpidosta tarkistaa, että vastaava ongelma on esiintynyt jo aiemminkin. Jokaisesta kommentista pitäisi jäädä myös aikaleima. Tällä ominaisuudella yritetään tarjota kirjaamismahdollisuus hiljaiselle tiedolle, siis tiedolle, joka muuten jäisi todennäköisesti yhden IT-henkilön omaksi muistikuvaksi ja unohtuisi ajan myötä.

Leasing-kohteiden, laskujen ja laitteiden suhde on erityinen – ne muodostavat eräänlaisen ketjun. Laskulla ei ole välttämättä aina leasing-kohdetta, muun muassa lisenssin sisältävillä laskuilla ei useinkaan ole leasing-kohdetta. Ketjumuodostelma leasing-kohteesta laitteeseen on esitetty kuvassa 5.2. Mukaan on otettu myös lisenssi.



Kuva 5.2 *Yksinkertaistettu kuva leasing-kohteesta lisenssiin tai laitteeseen muodostuvasta ketjumuodostelmasta.*

Kuvassa 5.2 olevat lukumääräsuhteet kertovat, että yksi lasku voi sisältää vain lisenssejä tai laitteita, ei molempia. Lisenssin on oltava suhteessa yhteen laskuun, samoin laitteen on kuuluttava yhteen laskuun. Leasing-kohde joko on tai ei ole olemassa, yksi lasku voi kuulua vain yhteen leasing-kohteeseen, mutta yhdelle leasing-kohteelle voi puolestaan kuulua useita laskuja. Kuva 5.2 on yksinkertaistettu esitys

neljän kuvassa olevan käsitteen suhteesta toisiinsa, sillä ei ole tarkoitus kuvata käsitteiden liittymistä muuhun käsiteympäristöön.

Lisensseillä ja laitteilla on tilatietoja, siis jokin staattinen tila, joka kertoo käsiteltävän objektin tilasta jotakin. Tilatiedoissa tuli ottaa huomioon kuvassa 5.3 esitetyt rajaehdot.

Kuva 5.3 *Lisenssien ja laitteiden tilatiedot määrittelevä vaatimuskuva.*

Lisenssien tilatieto voi olla tietyllä ajanhetkellä jokin neljästä kuvassa 5.3 esiintyneistä: se on voitu assosioida jollekin toimistolle, osastolle, henkilölle tai laitteelle. Laite voi puolestaan olla assosioituneena tietyllä ajanhetkellä jollekin seuraavista: toimisto, osasto tai henkilö.

Laitteiden ja lisenssien liikkeitä seurataan IT-hallintajärjestelmän tallentuvan historiatiedon pohjalta, esimerkiksi se, mihin laitteeseen mikäkin lisenssi on asennettuna, nähdään järjestelmästä. Järjestelmän avulla seurataan myös verkkolisenssien käyttöä ja luodaan raportit lisenssien käytöistä. Raporttien avulla hoidetaan yrityksen sisäinen laskutus eri osastojen välillä.

5.2. Integraatioprojekti

Integraatioprojektin vaatimuksena oli saada siirrettyä vanhan Access-tietokannan tiedot uuteen järjestelmään. Tietokannan sisältö oli lähes pelkästään laitteita, sekä käytöstä poistuneita että käytössä olevia. Joiltain osin relaatioiden ominaisuudet tulisivat vastaamaan toisiaan ja siirto olisi helppoa tehdä suoraviivaisesti. Esimerkiksi laitteiden kalustonumeroita vastaava ominaisuus luultavasti löytyisi uudesta tietokantaratkaisusta, vanha kalustonumero olisi siten uudessa tietokannassa samanniminen ominaisuus kuin vanhassa.

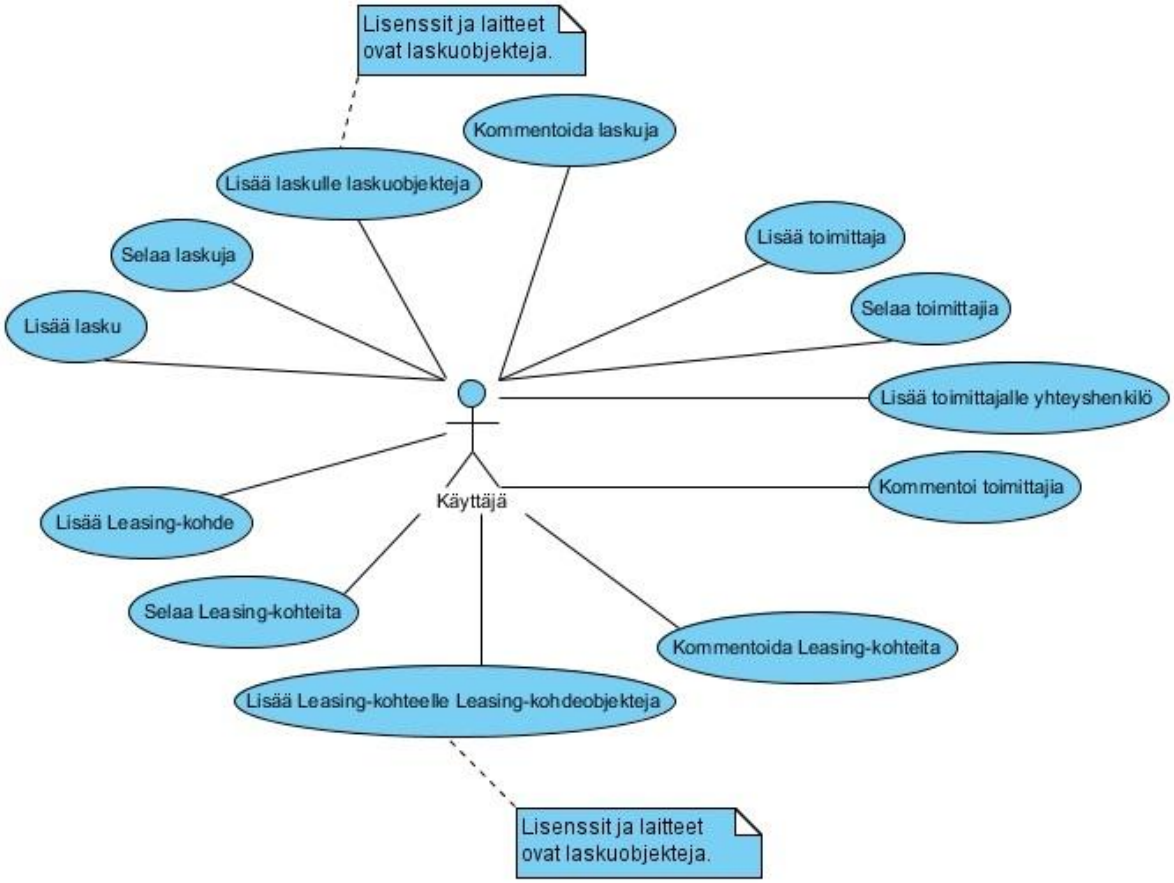
Jo etukäteen oli ilmeistä, ettei kaikkia vanhan järjestelmän ominaisuuksia olisi samassa merkityksessä uudessa järjestelmässä, vaikka samat tiedot sinänsä säilöttäisiin. Tällaisissa tilanteissa ei vanhoja tietoja voitu siirtää uuteen järjestelmään luonnollisille paikoilleen, siis vastaaviin ominaisuuksiin. Olisi kuitenkin tärkeää, ettei integraatioprojektin yhteydessä häviäisi mitään vanhan järjestelmän tietoa. Tässä voitiin käyttää laitteiden kommentointimahdollisuutta hyväksi ja sisällyttää esimerkiksi laitteen kommenttiin koko vanhan tietokannan raakadata.

5.3. Käyttöliittymä

Käyttöliittymän laadullisena vaatimuksena oli helppokäyttöisyys, sillä käyttöliittymää tultaisiin käyttämään hektisissä IT-osaston työtehtävissä ja käyttöliittymän tulisi olla

intuitiivisesti hyvällä tasolla. Käyttöliittymän suunnittelijan vastuulle jäi intuitiivisyysvaatimuksen lisäselvitys.

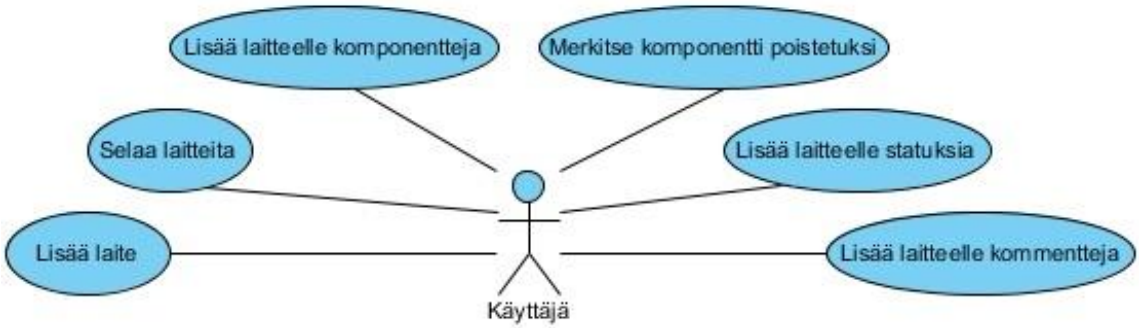
Käyttöliittymän toiminnalliset vaatimukset on helpointa käydä läpi käyttötapauskaavioiden avulla. Käyttöliittymän tulee jakaantuu pääosin kuvassa 5.1 olevien tietokannan pääosasien mukaan. Kuvassa 5.4 on esitetty laskuihin, leasing-kohteisiin ja toimittajiin liittyvät käyttötapaukset.



Kuva 5.4 *Käyttötapauskaavio liittyen laskuihin, leasing-kohteisiin ja toimittajiin.*

Laskuja on pystyttävä lisäämään, selaamaan ja kommentoimaan. Laskuihin pitää voida jälkikäteenkin lisätä niin sanottuja laskuobjekteja, joita ovat lisenssit ja laitteet. Samoin leasing-kohteita on voitava lisätä, selata ja kommentoida. Niille tulee lisäksi pystyä lisäämään niin sanottuja leasing-kohdeobjekteja eli laskuja. Toimittajia on samoin voitava lisätä, selata ja kommentoida. Lisäksi toimittajille pitää pystyä lisäämään uusia yhteyshenkilöitä toimittajan luomisen jälkeenkin. Toimittajien lisäyksen yhteydessä järjestelmälle pitää voida kertoa, onko kyseinen taho laitteiden, lisenssien, palveluiden vai joidenkin edellä mainittujen yhdistelmän toimittaja.

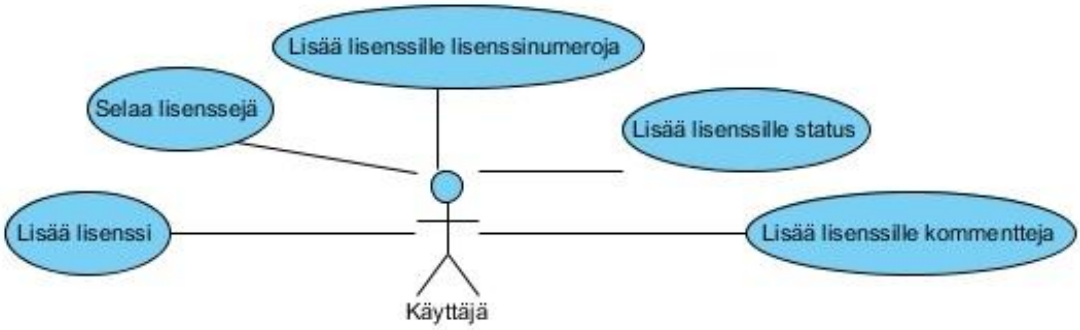
Laskuihin liittyvät kiinteästi lisenssit ja laitteet. Yhdellä laskulla voi olla useampia laitteita. Laitteiden käyttötapauskaavio on esitetty kuvassa 5.5.



Kuva 5.5 *Käyttötapauskaavio kuvaa laitteisiin liittyviä vaatimuksia.*

Laitteita pitää voida lisätä, selailla ja kommentoida. Lisäksi niiden tilatietoa pitää voida päivittää ja niiden komponentteja pitää voida lisätä ja poistaa, toisin sanoen merkitä, ettei kyseinen komponentti enää jostain päivämäärästä eteenpäin kuulu laitteen kokoonpanoon. Tietokannastahan ei vaatimusten mukaan pitänyt voida poistaa mitään tietoa käyttöliittymän kautta.

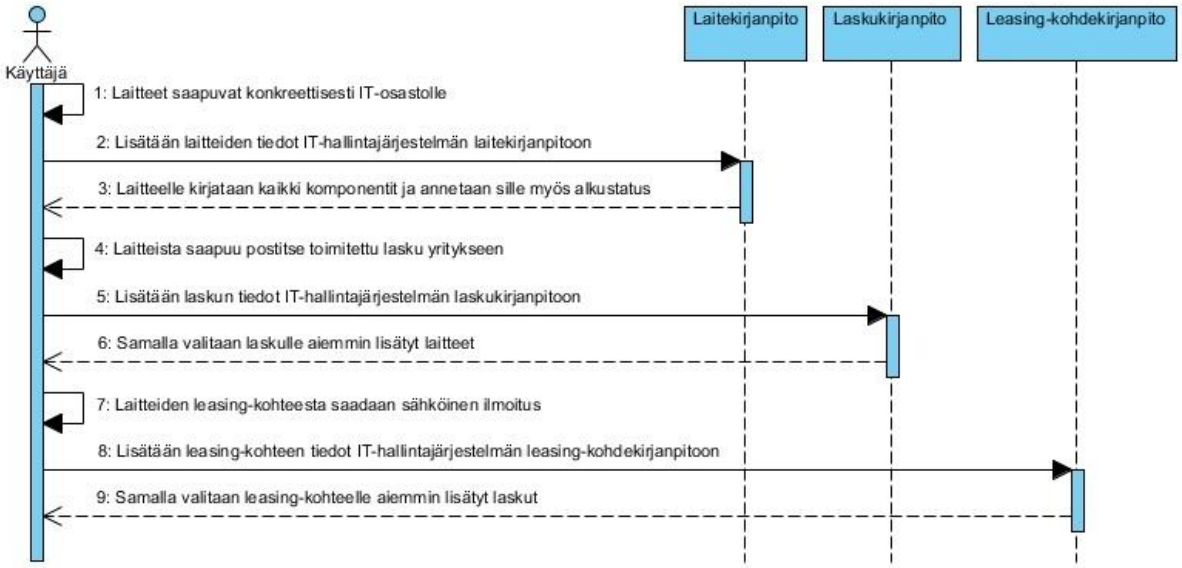
Laitteiden kanssa samalla tasolla laskuihin nähden ovat lisenssit. Lisensseihin liittyvä käyttötapauskaavio on esitetty kuvassa 5.6.



Kuva 5.6 Käyttötapauskaavio kuvaa lisensseihin liittyviä vaatimuksia.

Myös lisenssejä pitää pystyä lisäämään, selaamaan ja kommentoimaan. Lisäksi niille pitää voida lisätä tilatietoja ajan mittaan. Lisenssien erikoisuus on lisenssinumeroiden muuttuminen, ja lisenssinumeroita onkin voitava lisätä ajan myötä.

Leasing-kohteiden, laskujen ja laitteiden ollessa yhteydessä toisiinsa ketjun tavoin ei ole yhdentekevää, missä järjestyksessä niitä voidaan lisätä. Oheinen kuva 5.7 esittää sekvenssikaaviota siitä, missä järjestyksessä laitteet, laskut ja leasing-kohdetiedot todellisuudessa saapuvat yritykseen, ja missä järjestyksessä niitä pitää IT-hallintajärjestelmän kannalta käsitellä.



Kuva 5.7 *Sekvenssikaavio laitteiden, laskujen ja leasing-kohteiden lisäysjärjestysvaatimuksesta.*

Laitteet saapuvat yritykselle ennen kuin on mitään tietoa laskun tai leasing-kohteen tiedoista. Niille annetaan yksilöllinen laitenumero ja kirjataan muut laitteen identifioivat tiedot IT-hallintajärjestelmään. Keskusyksiköiden kohdalla pitää huolehtia siitä, että kaikki komponentit tulevat kirjatuksi. Lasku saapuu vasta laitteiden jälkeen ja se ajetaan järjestelmään. Samalla sidotaan siihen kuuluvat laitteet laskulle. Vasta viimeisenä saadaan leasing-kohteen tiedot, jotka voidaan lisätä järjestelmään samalla kirjaten sille kuuluneet laskut leasing-kohteelle.

6. TOTEUTUS

6.1. Muut markkinoilla olleet IT-hallintajärjestelmät

Aluksi kartoitettiin muita vastaavia järjestelmiä ja niiden tarjoamia ominaisuuksia. Markkinoilla oli useampia hallintajärjestelmiä, jotka pystyivät suoriutumaan erinomaisesti laitteiden ja jopa laskujen kirjanpidosta. Lisenssien hallinnan puute oli jokaisen tutkitun järjestelmän puute. Lisenssien hallinnassa vaatimusten mukaan oleellista oli saada sekä kiinteät yksittäislisenssit että verkkolisenssit hallintaan niin, että kirjanpitotietojen pohjalta onnistuu myöhemmin kustannuspaikkakohtainen laskutus.

Päädyttiin yhteistuumin toimeksiantajayrityksen kanssa siihen, että aloitetaan oman kyseiseen tarpeeseen räätälöidyn ratkaisun suunnittelu. Erityisesti käyttötarkoitukseensa suunnitellun hallintajärjestelmän voi tarpeen tullen mukauttaa uusien vaatimusten mukaiseksi helpommin, nopeammin ja täsmällisemmin kuin ulkopuolisen suunnitteleman.

6.2. Tietokanta

6.2.1. Käsitteellinen suunnittelu

Tietokannan suunnittelu alkoi käsitteellisellä suunnittelulla. Vanhan järjestelmän tietokanta toimi hyvänä lähtökohtana ja sen pohjalta voitiin miettiä tulevaa uutta järjestelmää. Vaatimusten mukaan oli selvää, että vanhassa järjestelmässä ei ole mitään turhaa tietoa, päinvastoin kaikki tiedot vanhasta järjestelmästä tuli sisällyttää tavalla tai toisella myös uuteen järjestelmään. Lisäksi tarvittiin paljon lisää tietoa ja etenkin juuri tietojen jäsentelyyn tuli kiinnittää paljon huomiota. Lisätiedot kerättiin IT-henkilöitä haastattelemalla.

Tietokantaa miettiessä oli alusta pitäen selvää tehdä siitä heti UML:n mukainen luokkakaavionotaatiota noudattava kuvaus, josta kävisi selkeästi ilmi taulut ominaisuuksineen ja taulujen väliset lukumääräsuhteet. Tällä tavalla valmis kaavio olisi selkeä ja helposti hallittava. Tampereen teknillisen yliopiston Ohjelmistotekniikan laitoksella järjestetyn Tietokannan suunnittelu -kurssin oppien mukaan kaaviosta lähdettiin jäsentämään tekstuaalista esitystä tietokantaskripteistä, joilla tietokanta tultaisiin luomaan.

6.2.2. Looginen suunnittelu

Tavoitteena oli siis saada tietokantaa kuvaava käsitekaavio muokattua tietokannan täydellisesti kuvaavaan tekstitiedostoon. Tämä tekstitiedosto voitaisiin ajaa

tietokannanhallintaohjelmalla ja lopputuloksena olisi syntynyt juuri halutun kaltainen tietokanta.

Ennen skriptin kirjoitusta oli tehtävä looginen suunnittelu, eli koko käsitekaavio muutettiin käännösprosessin avulla tekstimuotoisiksi relaatioiksi. Tässä pyrittiin kirjallisesti kertomaan samat asiat yhtä täydellisesti kuin ne olivat käsitekaaviossa. Käsitekaaviossa olevien lukumääräsuhteiden mukaan piti päätellä, onko relaatioiden välistä suhdetta varten luotava oma liitostaulunsa. Monesta moneen -suhteisiin luotiin liitostaulut, joilla eri taulujen tiedot liitettäisiin toisiinsa. Tässä loogisen suunnittelun vaiheessa valittiin myös taulujen pääavaimet. Liitostauluja lukuun ottamatta kaikissa tauluissa käytettiin surrogaatteja eli keinotekoisesti luotuja identifiointitunnuksia.

Tässä vaiheessa tarkistettiin vielä, että tietokannan normalisointi oli kunnossa, eli ettei esimerkiksi löytynyt turhaa toisteisuutta ja että kaikki tiedot olivat atomisia. Tietokanta pyrittiin mahdollisuuksien mukaan saamaan aliluvussa 3.2.3 esiteltyyn kolmanteen normaalimuotoon (3NF).

6.2.3. Fyysinen suunnittelu

Fyysisessä suunnittelussa luodaan lopulliset tietokannan luontiin käytettävät skriptit ja määritellään käyttäjien näkymät. Skriptit toimivat vain tietyn tietokannanhallintajärjestelmän kanssa, kun taas aiemmissa vaiheissa tehdyt tietokantakuvaukset ovat olleet riippumattomia tietokannanhallintajärjestelmästä.

Taulujen luomisen lisäksi tietokannan rakentamiseen kuului paljon muitakin toimia, kuten esimerkiksi kiinteiden tiedossa olevien tietojen lisääminen tauluihin, tallennettujen proseduurien luominen ja niin edelleen. Kukin osa-alue oli fyysisesti oma skriptitiedostonsa, joita voitiin sitten kootusti ajaa päätason skriptitiedoston kautta. Tällä tavalla skriptit olivat hyvin jäsennettyjä ja tiedostojen pituudet kohtuullisia. Niiden käsittely oli helpompaa, kun ne olivat selkeästi omia kokonaisuuksiaan.

6.3. Integraatioprojekti

Tässä työssä merkittävää oli vanhan Access-tietokannan tietojen siirtäminen uuteen järjestelmään samalla muuttaen niiden jäsentelyä. Tietojen siirto voitaisiin tehdä Microsoft SQL Integration Services (SSIS) -projektilla. SSIS-projekti suunniteltiin Microsoft Visual Studio Business Intelligence -ohjelmalla. Tietojen siirtoon liittyvät debuggausraportoinnit tehtiin samalla työkalulla. Debuggausraportoinnilla tässä yhteydessä tarkoitetaan joidenkin vanhojen tietojen suoraa yhteensopimattomuutta uuden järjestelmän kanssa. Näitä tietoja varten tehtiin omia debuggaustekstiraportteja, joiden pohjalta SSIS-projektia voitiin parantaa siten, että kaikki tieto saataisiin siirrettyä uuteen järjestelmään. Varsinainen lopullinen tiedon siirto eli SSIS-projektin suoritus hoidettiin Visual Studiolla Business Intelligence -ohjelmalla, koska projekti oli kertaluontoinen eikä sitä tarvinnut ajastaa säännöllisesti ajettavaksi. Samaisella ohjelmalla tullaan jatkossa luomaan Microsoft Reporting Services -raporttipohjat.

SSIS-projekti jakaantui yhtenäiseen kontrolliketjuun, joka saneli missä järjestyksessä mikäkin toimenpide tehtiin. Kukin toimenpide oli oma kokonaisuutensa, joka sisälsi esimerkiksi tietojen hakemista eri tauluista, tiedon jatkojalostusta ja lopulta sen tallentamista uuteen tietokantarakenteeseen.

Sen lisäksi, että uuteen järjestelmään tuotiin tietoa vanhasta järjestelmästä käyttämällä SSIS:ää, sillä luotiin myös täysin uutta tietoa. Osa vanhoista tiedoista oli määritelty liian yleisellä tasolla, minkä vuoksi joukossa oli vanhentunutta tietoa, esimerkiksi käyttäjiä, joita enää ei ollut yrityksen palveluksessa. Näitä erikoisia poikkeustilanteita varten tietojen tuontiin jouduttiin kehittämään ehtoja, jotka muuttavat vanhan tuotavan tiedon käyttökelpoiseksi uuteen järjestelmään. Osa vanhan järjestelmän tiedoista korjattiin suoraan Access-tietokantaan, jotta vältyttäisiin turhalta työltä siirto-operaation yhteydessä.

6.4. Käyttöliittymä

6.4.1. Käyttöliittymän suunnittelu

Koska suunnittelutyöhön oli käytettävissä rajallisesti resursseja, työn eri osa-alueita tuli priorisoida. Tietokannan suunnittelu ja integraatioprojekti saivat siksi suuremman huomion kuin käyttöliittymäsuunnittelu. Teoriaosuuden aliluvussa 3.5 esiteltyjä suunnitteluperiaatteita kuitenkin pyrittiin noudattamaan pääpiirteittäin. Käyttöliittymää suunniteltiin suurelta osin Bottom-Up -periaatteella, eli pienemmistä käytännön yksityiskohdista edettiin kohti suurempia kokonaisuuksia. Lopulta pystyttiin katsomaan kokonaisuutta ylhäältä alaspäin, kun käyttöliittymä oli riittävän valmis.

Käyttöliittymän suunnittelu ja ohjelmointi etenivät samanaikaisesti. Aliluvussa 5.3 esitettyjen käyttötapauskaavioiden mukaan luotiin käyttöliittymän välilehdet. Pääosassa välilehdillä oli listaus objekteista ja listaukseen liittyvä suodatin. Näiden lisäksi kaikille välilehdille laitettiin lisäys-nappulat.

Käyttöliittymän helppokäyttöisyys oli laadullisista vaatimuksista merkittävin, mikä selvisi haastattelemalla tulevia käyttäjiä toivotuista käyttötavoista ja rutiineista.

6.4.2. Käyttöliittymän toteutus

Ohjelmointi suoritettiin käytännössä käyttötapausten pohjalta, koska tarkkoja suunnitelmia ei ollut. Ohjelmoinnin testaukseen pyrittiin keskittymään mahdollisimman paljon, testaus suoritettiin peilaamalla valmista järjestelmää käyttötapausten vaatimuksiin.

7. TULOKSET JA NIIDEN ANALYSOINTI

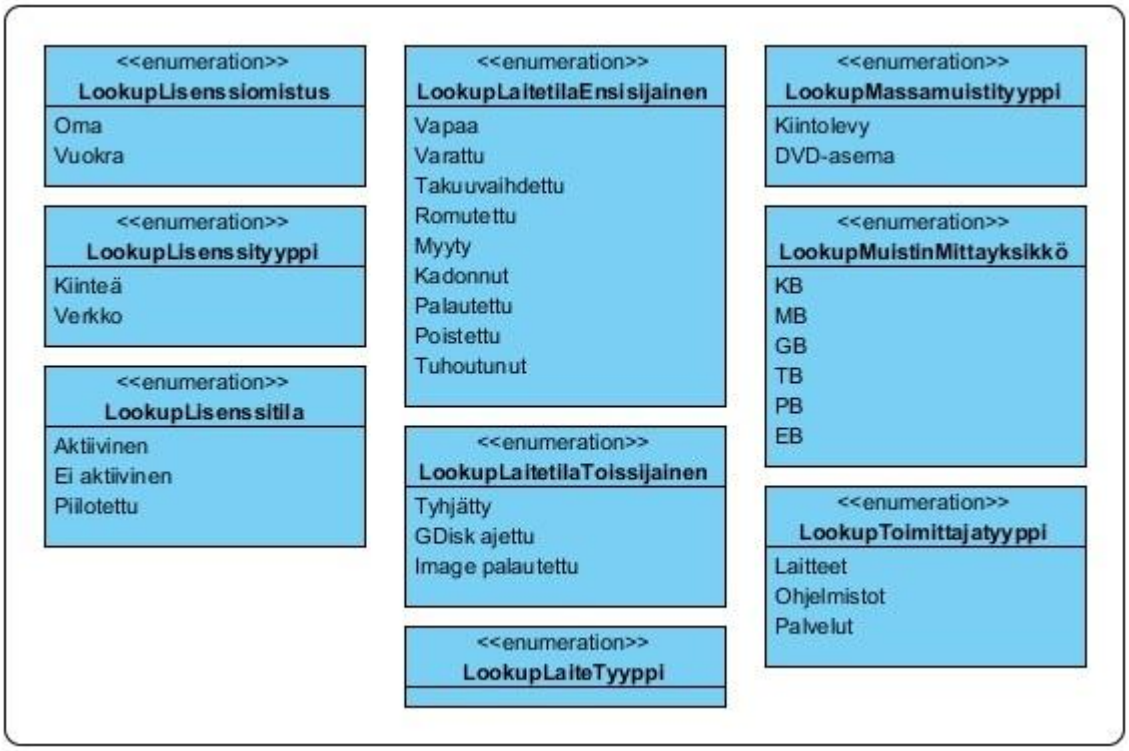
7.1. Tietokanta

7.1.1. Tietokannan relaatiokaaviot

Lähtötietojen ja vaatimusten perusteella päädyttiin jakamaan relaatiokaavio isompiin osakokonaisuuksiin. Päälohkot muodostettiin kuvan 5.1 mukaisesti. Liitteessä 1 on esitetty yhtenä kuvana koko käsitekaavio, mutta tässä aliluvussa käydään käsitekaavio paloittain läpi. Päälohkojen lisäksi luotiin luokat, joita käytettiin lähes jokaisessa päälohkossa, useissa tauluissa. Nämä luokat on esitetty kuvassa 7.1.

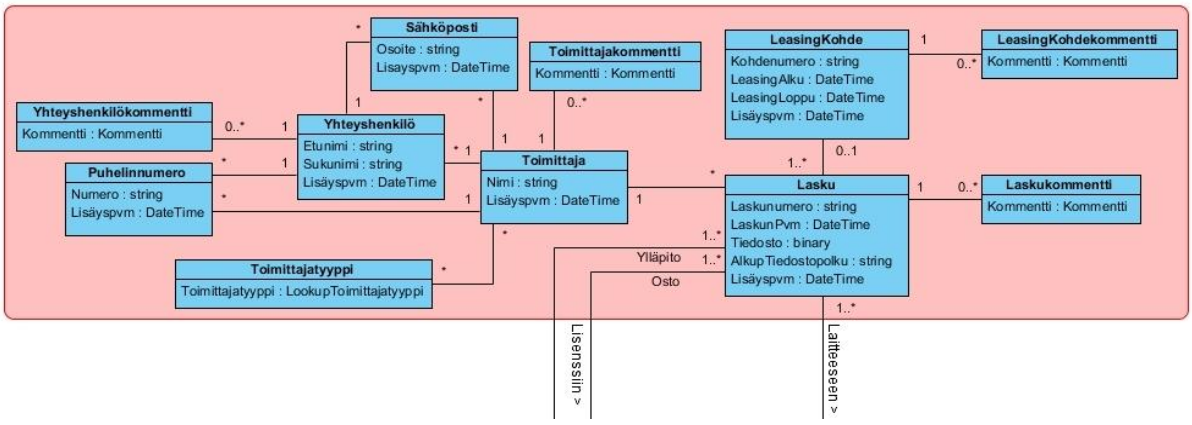
Kuva 7.1 *Kommentti ja sijainti -objektit erotettiin omaksi kokonaisuudekseen.*

Toinen erillinen lohko, jota ei vaatimuksissa osattu ottaa huomioon, oli Lookup-taulut. Lookup-taulut on esitetty arvoineen kuvassa 7.2, pois lukien laitteiden tyyppi, josta arvoja ei ole laitettu näkyviin niiden suuren määrän vuoksi. Lähes kaikki laitteiden tyypit on esitelty aiemmin taulukossa 4.2.



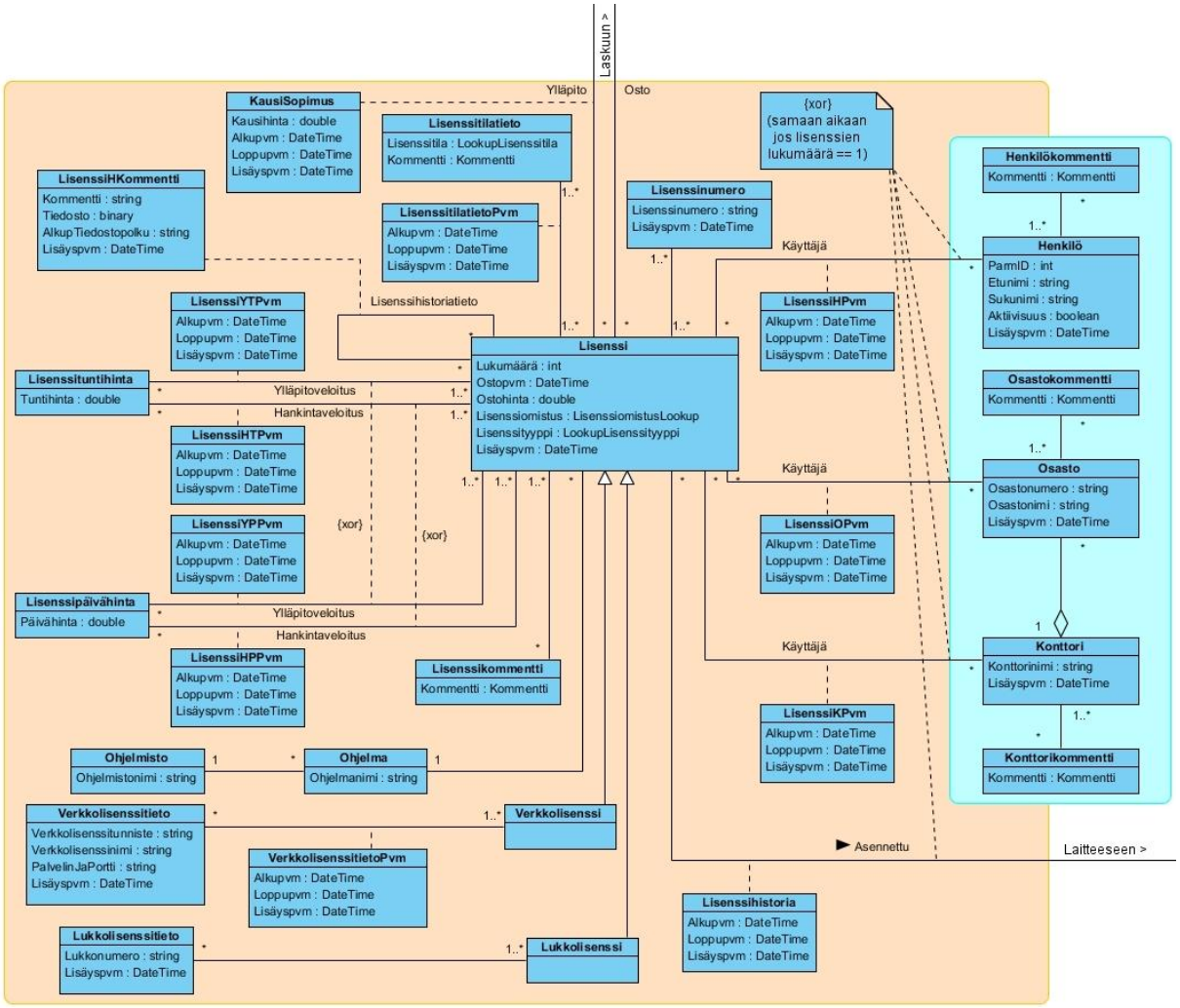
Kuva 7.2 *Lookup-taulut omana lohkonaan.*

Lookup-taulujen tietojen on tarkoitus olla stabiileja, mutta se ei tarkoita sitä, etteikö sinne voisi lisätä tai jopa poistaa jotakin, jos tarve niin vaatii. Lisäykset ja poistot on tarkoitus tehdä tietokannanhallintajärjestelmän kautta käsityönä. Lookup-tauluja käytetään muun muassa laskuja, leasing-kohteita ja toimittajia kuvaavassa lohkossa. Tämän lohkon tarkempi sisältö on esitettynä kuvassa 7.3.



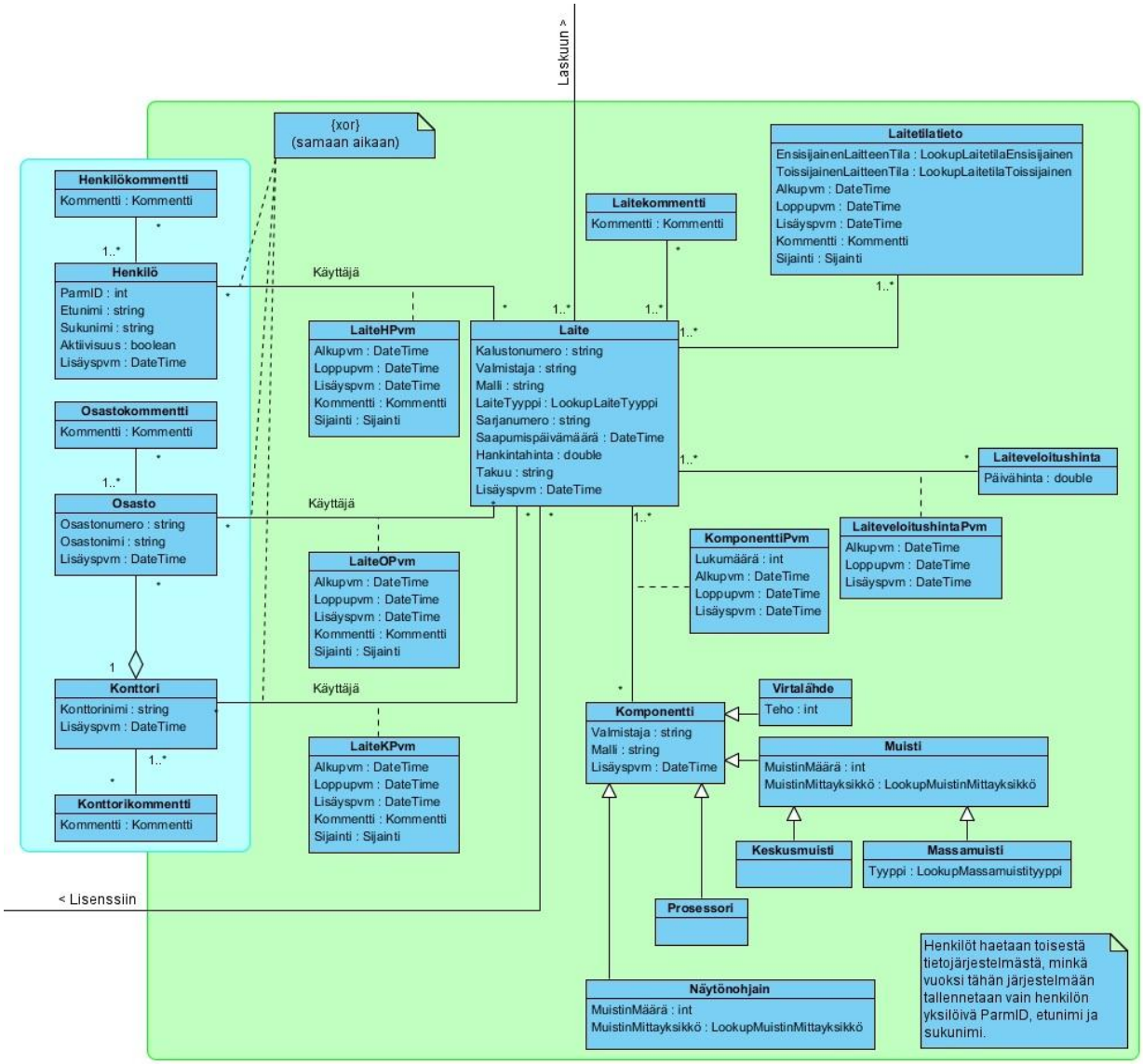
Kuva 7.3 *Laskut, leasing-kohteet ja toimittajat sisältävä lohko tarkemmin kuvattuna.*

Laskuista johtaa kaksi yhteyttä lisenssiin, toinen kuvaa lisenssin ylläpitolaskua ja toinen ostolaskua. Lisenssin ylläpitolaskuun liittyy myös käsite KausiSopimus, joka näkyy Lisenssi-lohkon tarkemmassa erittelyssä kuvassa 7.4.



Kuva 7.4 *Lisenssi-lohko tarkemmin eriteltynä.*

Lisenssi-lohkossa näkyy myös Käyttäjät-lohko, joka liittyy lisenssien lisäksi kiinteästi myös laitteisiin. Laitteet-lohkossa on samainen Käyttäjät-lohko esitetty uudestaan. Laitteet-lohko on eritelty tarkemmin kuvassa 7.5.



Kuva 7.5 *Laitteet-lohkon tarkempi erittely.*

Edellä esitettyjen lohkojen pohjalta voitiin kirjoittaa tietokantaskriptit, jotka käydään pääpiirteissään läpi seuraavassa aliluvussa.

7.1.2. Tietokantaskriptit

Tietokannan luomista varten kirjoitettiin useampia tietokantaskriptitiedostoja, joilla lopullisen tietokannan saattoi luoda parilla hiiren klikkauksella käyttäen SQL Server Management Studio -tietokannanhallintajärjestelmää. Lohkojaottelusta siirryttiin skriptitiedostojen osalta toimintojen tyypittelyyn, kuten taulujen luomiseen, tietojen lisäykseen ja niin edelleen. Taulukossa 7.1 on eritelty skriptitiedostot, jotka ajetaan yhdestä kootusta päätiedostosta.

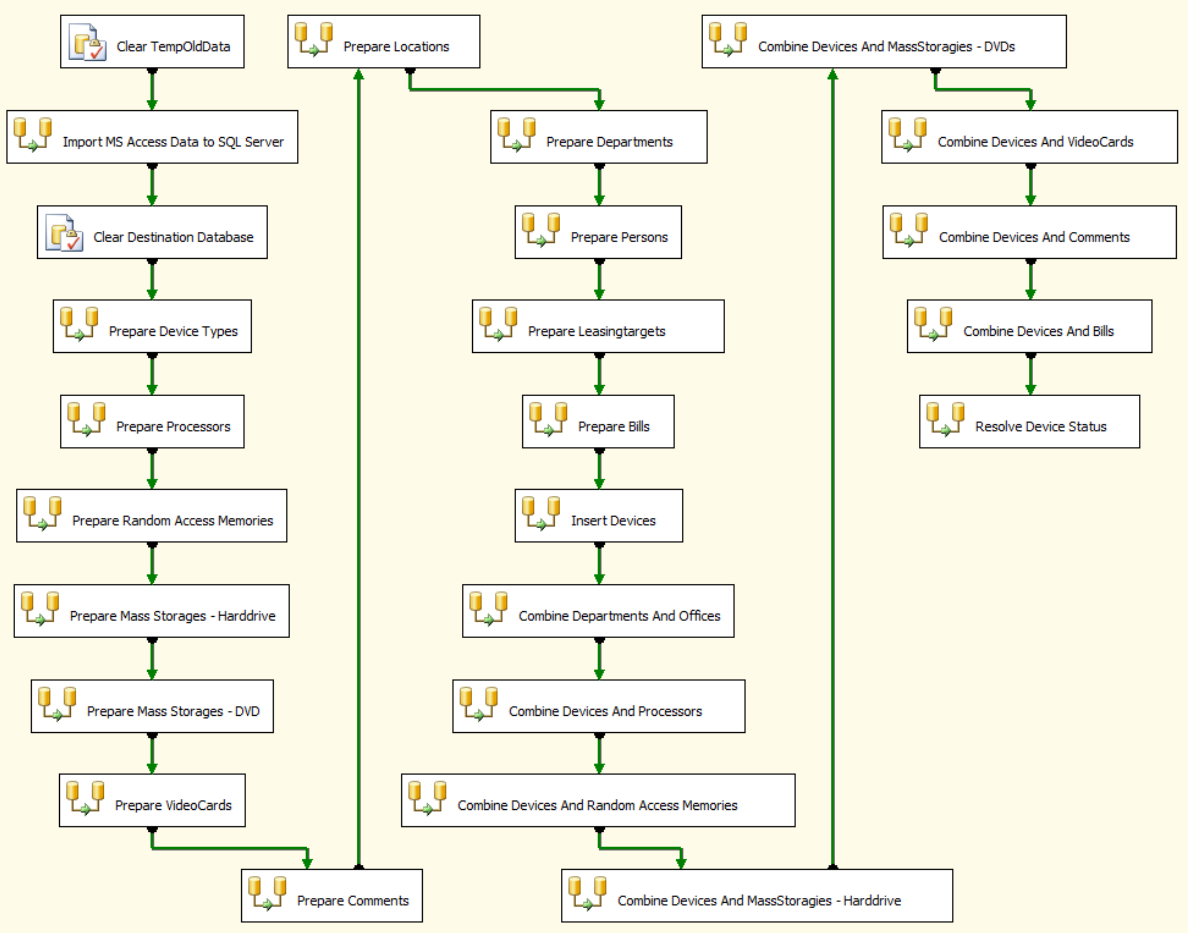
Taulukko 7.1 *Tietokantaskriptien jaottelu.*

Toiminto	Selitys
Tietokanta	Luodaan itse tietokanta muuta sisältöä varten.
Taulut	Luodaan tietokantaan taulut, joihin data tallennetaan.
Lookup-taulut	Luodaan niin sanotut lookup-taulut, joiden funktio on tallentaa staattista tietoa. Niistä luetaan tietoa erinäisiin alasvetovalikoihin ja valintalistoihin.
Oikeudet	Annetaan oikeuksia tietyille käyttäjille tiettyihin tauluihin, tallennettuihin proseduureihin ja muihin tarvittaviin toimintoihin tietokannan sisällä.
Näkymät	Luodaan näkymiä, joita käytetään tietokantaa luettaessa.
Lookup-data	Lisätään lookup-tauluihin ennalta määritelty data.
Taulu-data	Joihinkin tauluihin lisätään dataa myös ennakkoon, koska jälkikäteen sen lisääminen ei onnistu niin helposti.
Tallennetut proseduurit	Luodaan tallennettuja proseduureja, joilla nopeutetaan ja yksinkertaistetaan joitakin tietokantaan liittyviä toimintoja. Useimmat proseduurit ovat varsin monimutkaisia, mutta samojen asioiden tekeminen muilla tavoin olisi erittäin työlästä.
Väliaikaistaulut	Luodaan niin sanottuja väliaikaistauluja, jotta tietojen siirto vanhasta Access-tietokannasta uuteen järjestelmään sujuisi mahdollisimman mutkattomasti.

Kaikkiaan tietokannan luomiseen vaadittavista skripteistä muodostui yhteensä suuri määrä SQL-komentoja. Yhteenlaskettuna tiedostoissa on lähes 2000 riviä SQL-komentoja, jotka kuvaavat tarkasti sen, millä tavalla tietokanta kuuluu luoda.

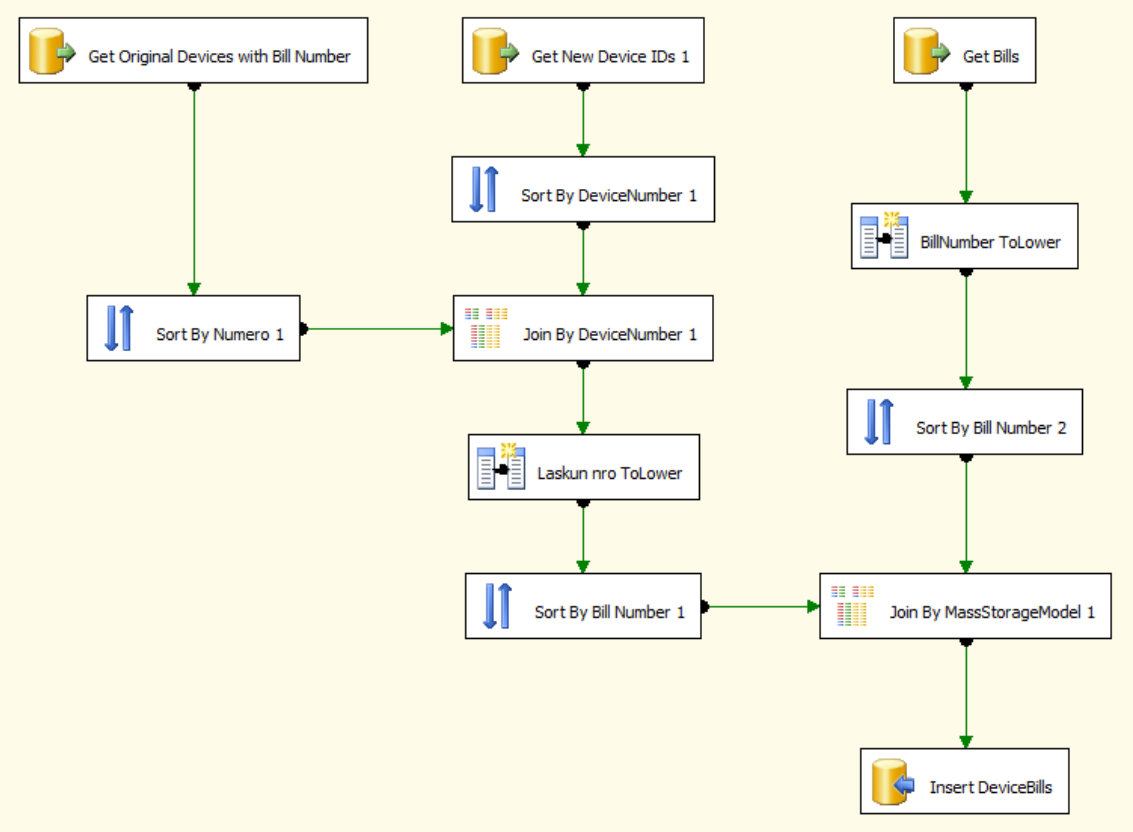
7.2. Integraatioprojekti

Tietokannan luonnin jälkeen oli ajettava vanhan datan siirto uuteen. Tätä varten luotiin SQL Server Integration Services -projekti, jota tässä tapauksessa ajettiin suoraan Microsoft Visual Business Intelligence Studio -kehitystyökalulla kertaluontoisesti. SSIS-projektin kehitys tapahtui graafisesti, mutta tekstimuotoisesti tarkasteltuna määrittelytiedosto yli 30 000-rivinen järkäle. Kuvassa 7.6 on esitetty graafisen kehitystyökalun projektin Control Flow, joka ylimmällä tasolla kertoo, mitä tehtäviä projektiin kuuluu, ja mikä on niiden suoritusjärjestys.



Kuva 7.6 *Kuvakaappaus integraatioprojektin Control Flow'sta.*

Kuvasta 7.6 voidaan havaita, kuinka integraatioprojekti jakautuu muutamaan pääosaan: väliaikaistietokannan toimet, varsinaisen tietokannan taulujen valmistelu, laitteiden vienti ja lopuksi yhdistelytoimet laitteiden ja muiden tietojen osalta. Kukin Control Flow:n tehtävä sisältää kokonaisuuden, josta esimerkkinä kuvassa 7.7 oleva Data Flow tehtävästä Combine Devices And Bills.

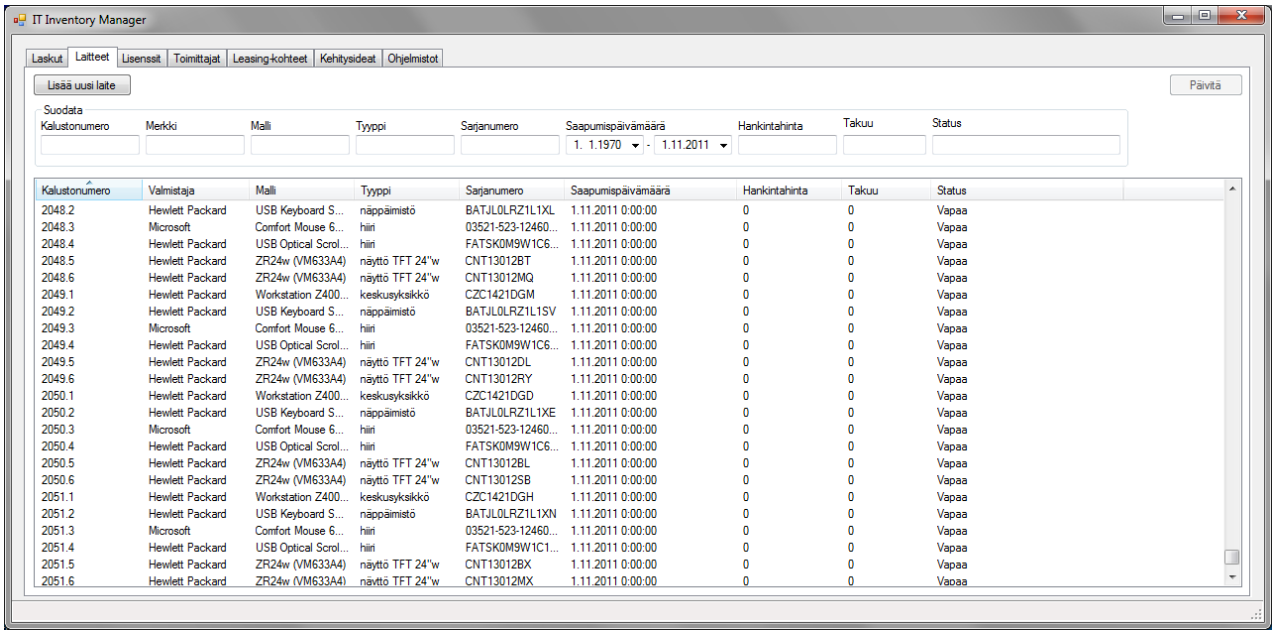


Kuva 7.7 *Control Flow'n Combine Devices And Bills -tehtävän sisältö.*

Combine Devices And Bills -tehtävän sisällössä on kolme tiedonhakuoperaatiota ja yksi tiedon vientioperaatio lopussa. Välissä tietoja muokataan ja yhdistellään, jotta saadaan haluttu lopputulos.

7.3. Käyttöliittymä

Käyttöliittymästä muodostui ehyt kokonaisuus, eikä sen ohjelmoinnissa törmätty suuriin ongelmiin. Oheisessa kuvassa 7.8 on kuvakaappaus käyttöliittymästä, Laitteet-välilehdestä.



Kuva 7.8 *Kuvakaappaus käyttöliittymän Laitteet-välilehdestä.*

Kuvakaappauksessa näkyy listaus laitteista. Listan yläpuolella on suodin, jolla voidaan erittäin helposti ja nopeasti suodattaa listanäkymä mieleiseksi. Yläreunassa näkyvät muutkin käyttöliittymän välilehdet, jotka on eritelty tehtävineen taulukossa 7.2.

Taulukko 7.2 *Käyttöliittymän välilehtien merkitykset.*

Välilehti	Merkitys
Kehitysideat	Tämän välilehden kautta voidaan lisätä tietokantaan ja koodaajan tiedoksi kehitysideoita. Välilehden tarkoitus on siis kehityskeskeinen.
Laitteet	Laitteiden listaus on tällä välilehdellä, kustakin laitteesta on perustietojen lisäksi listattu viimeinen tilatieto. IT-henkilöiden töissä tarvitaan useimmiten juuri tilatietoa.
Laskut	Laskujen listaus ja lisäys tehdään tätä kautta. Laskuista on listattu kaikki oleellinen tieto.
Leasing-kohteet	Kaikki järjestelmässä olevat leasing-kohteet listataan tällä välilehdellä. Niitä tuplaklikkaamalla päästään tutkimaan leasing-kohteen laitteita ja lisenssejä.
Lisenssit	Lisenssit listataan tällä välilehdellä. Erityisen tärkeää on lisenssin viimeisin tilatieto.
Ohjelmistot	Tämän välilehden tehtävänä on listata ohjelmistot ja ohjelmat. Ohjelma listaus riippuu valitusta ohjelmistosta.
Toimittajat	Tällä välilehdellä on listaus laite- ja lisenssitoimittajista sekä uuden toimittajan lisäys.

Käyttöliittymän ohjelmointi vaati vähäisestä priorisoinnista huolimatta eniten työtunteja, ja lopputuloksena muodostui yli 40 000 koodiriviä C#-ohjelmointikoodia IDE:n automaattisesti luodut tiedostot mukaan luettuna. Ohjelmointikoodia sisältävien tiedostojen lukumäärä kohosi yli 200:an.

7.4. Työn arviointi

Luvussa 5 esitetyt vaatimukset saavutettiin, ja työtä voidaan siten pitää varsin onnistuneena. Vaatimuksiin eivät kuuluneet luotavan järjestelmän säilömistä tiedoista kootut raportit, mutta uuden hallintajärjestelmän hyöty saadaan täysimääräisenä irti, kunhan raportit on luotu.

Ennalta määritelty aikataulu ei pitänyt. Alkuperäisen toimeksiannon mukaan työ piti saada puolessa vuodessa valmiiksi. Työn suoritukseen kului kuitenkin melkein puolitoista vuotta.

8. JATKOKEHITYS

8.1. Käyttäjäryhmät

Järjestelmän käyttäjähallinta oli alun perin tarkoitus hoitaa SQL-tietokannan käyttöoikeuksien avulla käyttäen hyväksi Windowsin käyttäjäryhmiä. Alkuvaiheessa tuotiin esiin ajatus, että käyttöliittymän käyttäjäryhmiä olisi kolme. Kutakin varten luotaisiin oma Windows-käyttäjäryhmä. Ryhminä olisivat raporttikäyttäjät, ylläpitokäyttäjät ja pääkäyttäjät. Näistä viimeisimpään ryhmään kuuluisi luonnollisesti vain yksi tai kaksi ihmistä.

Raporttikäyttäjällä olisi ollut oikeus ainoastaan lukea tietoja ja tulostaa raportteja. Tähän käyttäjäryhmään olisi voitu ajatella kuuluvan esimerkiksi yrityksen osastojohtajat, jotka haluavat tarkastella oman yksikkönsä kustannuskertymää IT-laitteiden ja lisenssien osalta. Heille olisi myös ennakkoon räätälöity tiettyjä heidän omaa toimintaansa tukevia raportteja.

Ylläpitokäyttäjät olisivat päässeet tekemään kaikki samat asiat kuin raporttikäyttäjät, mutta heille olisi myös sallittu tiettyjen tietojen lisääminen. Ylläpitokäyttäjä voisi esimerkiksi lisätä laitteelle uuden käyttäjän ja sijainnin.

Pääkäyttäjillä olisi ollut käyttäjistä eniten oikeuksia. Heidän olisi mahdollista tehdä samat asiat kuin raporttikäyttäjän ja ylläpitokäyttäjän, minkä lisäksi heidän olisi mahdollista muun muassa muuttaa jo olemassa olevaa tietoa (toisin sanoen korjata ja poistaa virheellistä tietoa).

Nykyisessä järjestelmässä juuri ylläpitokäyttäjien osuus on toteutettu, heillä on mahdollisuus tehdä kaikki tarvittavat ylläpitotoimenpiteet toteutetun käyttöliittymän kautta. Raporttikäyttäjien osuus päätettiin toteuttaa SQL Server Reporting Service -palvelun avulla. Olisi harkitsemisen arvoista toteuttaa ylläpitokäyttäjien osuus nykyisellä käyttöliittymällä, mutta toistaiseksi se tehdään käsin tietokannanhallintaohjelman kautta.

8.2. Tietokantatarkennuksia

Tietokannan taulujen ja taulujen ominaisuuksien tarpeellisuus ja riittävyys pitää harkita jonkin ajan kuluttua käyttöönotosta: olisiko tarvetta lisätä jotain, tai ovatko jotkin taulut tai taulujen ominaisuudet täysin turhia?

9. JOHTOPÄÄTÖKSET

Tämän opinnäytetyön merkittävin hyöty on uusi IT-laitteiden ja ohjelmistolisenssien hallintajärjestelmä, jonka suunnittelu ja toteutus on tehty harkiten. Myös laadullisesti hyvään ja samalla riittävään dokumentointiin on panostettu. Työ jakautui kolmeen osaan: tietokannan suunnitteluun, vanhojen tietojen viemiseen uuteen tietokantaan ja käyttöliittymän toteuttamiseen.

Työn aihealue oli melko haastava ja aiheeseen liittyvää lähdemateriaalia oli siksi vaikea löytää. Sen enempää kirjallisuudesta kuin internetistäkään ei löytynyt juuri IT-laitteiden ja ohjelmistolisenssien hallintaongelmaa käsittelevää materiaalia. Tämä toisaalta osoittaa työn hyödyllisyyden.

Työn tuloksena luotu uusi hallintajärjestelmä on suositeltavaa ottaa mahdollisimman pian käyttöön. IT-laitteiden ja ohjelmistolisenssien tietoja pystytään jatkossa käsittelemään joustavasti useamman ihmisen toimesta yhtäaikaisesti. Lisäksi kehitetyn järjestelmän avulla pystytään toteuttamaan osastokohtainen sisäinen laskutus, mikä tavoitteen mukaan johtaa siihen, että osastojen IT-kulut pysyvät paremmin kurissa. Osastopäälliköiden on helpompi tarkkailla IT-kulujen kohdentumista oman osastonsa sisällä, ja mahdollisiin ongelmakohtiin pystytään puuttumaan aiemmassa vaiheessa.

Järjestelmä vaatii varmasti jatkokehitystä. Harva ohjelmisto tai tietokantasuunnitelma on ensiversiolta täydellinen. Tässä tapauksessa hyvän dokumentoinnin toivotaan auttavan jatkokehitystoimissa.

LÄHTEET

[1] Insinööritoimisto Comatec Oy, Toimialat [WWW]. [Viitattu 30.10.2011]. Saatavissa: http://www.comatec.fi/fi/toimialat/

[2] SQL Server Database Engine. [WWW]. [Viitattu 1.11.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms187875(v=SQL.90).aspx

[3] The .NET Framework stack [WWW]. Orisoft Ltd. [Viitattu 16.11.2011]. Saatavissa: http://www.orisoft.co.uk/Content/themes/base/images/dot-net-stack.png

[4] Salmela Eila Helena. Tietokantojen historia ennen SQL:ää [WWW]. Tietojenkäsittelytieteen laitos. Helsinki, 4.5.2007. [Viitattu 2.11.2011]. Saatavissa: http://www.cs.helsinki.fi/u/kerola/tkhist/k2007/alustukset/tietokannat/TiKaHistoria.pdf

[5] Silberschatz Abraham, Korth Henry, Sudarshan Shrihari. *Database System Concepts*, Fifth Edition. 2005, McGraw-Hill Science. 1168 s.

[6] Coronel Carlos, Morris Steven, Rob Peter. *Database Systems: Design, Implementation, and Management*, Ninth Edition. 2009, Course Technology. 692 s.

[7] Peltonen Jari. Tietokantojen suunnittelu. 2007. Tieto- ja sähkötekninen tiedekunta. Vain kurssilaisille julkaistu luentomoniste. 106 s.

[8] SQL Server Integration Services. 2011. FC Sovelto Oy. Vain Sovellon kurssilaisille julkaistu luentomoniste. 124 s.

[9] Ahonen Markus. Tapaustutkimus: Soveltuuko Scrum vesiputousmallin korvaajaksi yrityksen sovelluskehitysprojekteihin? [WWW]. Elektroniikan, tietoliikenteen ja automaation tiedekunta, Automaatio- ja systeemitekniikan laitos. Espoo. 3.5.2010. [Viitattu 5.11.2011]. Saatavissa: http://lib.tkk.fi/Dipl/2010/urn100203.pdf

[10] Pohjonen, Risto. *Tietojärjestelmien kehittäminen.* Jyväskylä, 2002. Docendo. 178 s.

[11] Määttä Tiina, Rautio Pirkko. Toiminnallinen määrittely dokumentti ketterästi [WWW]. Rovaniemen ammattikorkeakoulu, Luonnontieteiden ala. 2010. [Viitattu 5.11.2011]. Saatavissa: https://publications.theseus.fi/bitstream/handle/10024/7469/maatta_tiina_rautio_pirkko.pdf?sequence=1

[12] Haikala Ilkka, Märijärvi Jukka. *Ohjelmistotuotanto,* 11. painos. 2006, Talentum. 440 s.

[13] Sommerville Ian. *Software Engineering,* Eight Edition. 2006. Addison Wesley. 864 s.

[14] Katara Mika. Ohjelmistojen testaus. [Luentomoniste]. Tieto- ja sähkötekniikan tiedekunta, Ohjelmistotekniikan laitos. [Viitattu 5.11.2011]. Saatavissa: http://www.cs.tut.fi/~testaus/s2011/luennot/